WEIGHTED MAHALANOBIS DISTANCE FOR HYPER-ELLIPSOIDAL CLUSTERING

THESIS

Khaled S. Younis
Captain, RJAF

AFIT/GE/ENG/96D-22

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

E

AFIT/GE/ENG/96D-22

WEIGHTED MAHALANOBIS DISTANCE FOR HYPER-ELLIPSOIDAL CLUSTERING

THESIS

Khaled S. Younis
Captain, RJAF

AFIT/GE/ENG/96D-22

AFIT/GE/ENG/96D-22

WEIGHTED MAHALANOBIS DISTANCE FOR HYPER-ELLIPSOIDAL CLUSTERING

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Khaled S. Younis, B.S.E.E

Captain, RJAF

December, 1996

Acknowledgements

At first, I thank God for giving me the ability to finish this research effort and to learn a little more about one small aspect of His universe. I would like also to thank the Royal Jordanian Air Force for giving me the great opportunity to study at the Air Force Institute of Technology.

This thesis could not have been done without the help of few individuals who deserve thanks and credit. First of all, I owe a great deal of thanks to my advisor, Dr. Steven Rogers, for his sharp insight, expertise, and enthusiasm. He provided me with plenty of rope, but still kept me from hanging myself. I thank him for the constant encouragement during the last few months. The comments of the other committee members, Dr. Martin DeSimio and Dr. Mathew Kabriski, have been extremely helpful to me in bringing this report to its finished form. Thanks to Dr DeSimio for providing friendly encouragement and many interesting discussions on new and bizarre clustering methods. I would like to thank my sponsor, Maj. Tom Burns from Wright Labs, for his support and guidance. I am also grateful for the network support provided so ably by Dan Zambon and Dave Doak.

I wish to acknowledge the support of the International Military Students Office and the assistance of Mr. Vic Grazier who helped me and other international students and our families since we arrived here.

Most importantly, I owe the greatest amount of gratitude to my wife, Maha. Her contribution to this project by her willing sacrifice of time and effort to support me may never be fully assessed. I feel lucky that God has chosen me to be your husband. Last but not least, I owe a special thanks for my beautiful daughter, Mira, and my son, Mostafa, for the joy that they add to my life. I hope that one day you will realize how many times I wanted to be with you but I couldn't.

Khaled S. Younis

Table of Contents

Page

List of Figures

Figure Page

List of Tables

AFIT/GE/ENG/96D-22

Abstract

Cluster analysis is widely used in many applications, ranging from image and speech coding to pattern recognition. A new method that uses the weighted Mahalanobis distance (WMD) via the covariance matrix of the individual clusters as the basis for grouping is presented in this thesis. In this algorithm, the Mahalanobis distance is used as a measure of similarity between the samples in each cluster. This thesis discusses some difficulties associated with using the Mahalanobis distance in clustering. The proposed method provides solutions to these problems. The new algorithm is an approximation to the well-known expectation maximization (EM) procedure used to find the maximum likelihood estimates in a Gaussian mixture model. Unlike the EM procedure, WMD eliminates the requirement of having initial parameters such as the cluster means and variances as it starts from the raw data set. Properties of the new clustering method are presented by examining the clustering quality for codebooks designed with the proposed method and competing methods on a variety of data sets. The competing methods are the Linde-Buzo-Gray (LBG) algorithm and the Fuzzy *c*-means (FCM) algorithm, both of them use the Euclidean distance. The neural network for hyperellipsoidal clustering (HEC) that uses the Mahalnobis distance is also studied and compared to the WMD method and the other techniques as well. The new method provides better results than the competing methods. Thus, this method becomes another useful tool for use in clustering.

WEIGHTED MAHALANOBIS DISTANCE FOR HYPER-ELLIPSOIDAL CLUSTERING

I. Introduction

1.1 Background

One of the most primitive and common activities of man consists of sorting like things into categories. The objective is to group either the data units (tanks, persons, images) or the variables (attributes, characteristics, measurements) into clusters such that the elements within a cluster have a high degree of "natural association" while the clusters are "relatively distinct" from one another.

Cluster analysis is very important in solving pattern recognition problems, especially when there are only unlabeled data points. Lacking the knowledge of the class membership of the data points, one cannot estimate the statistical properties of each class. Cluster analysis is also useful when the actual distribution of the sample is multimodal or can be characterized as a mixture of several unimodal distributions.

Many applications of pattern recognition and neural networks require clustering as a tool of learning and designing a pattern classifier based on the minimum distance concept. Unsupervised or self-organized learning algorithms have played a central part in models of neural computation. Examples are Hebbian learning, ART, and Kohonen feature maps [9]. Unsupervised models have been proposed as front ends for supervised learning problems [29]. Different approaches to clustering appear in Gersho and Gray [15], Bezdek [4], and Duda and Hart [12]. In spite of the numerous research efforts, data clustering remains a difficult and essentially an unsolved problem [18].

Clustering can be hard, where each data point is assigned to only one cluster, or soft, where clustering is basically a function assigning to each sample a degree of membership $u \in [0,1]$ to each cluster [7]. An example of the hard clustering is the algorithm proposed by Linde, Buzo, and Gray, commonly known as the LBG algorithm [24]. On the other hand, the most common algorithm in fuzzy clustering is the fuzzy c-means (FCM) outlined in Bezdek's paper [7].

One of the most important issues pertaining to clustering is the choice of the similarity measure. A form of the geometrical distance metric between points in the feature space is considered a logical choice. A general form of the distance between vectors $\boldsymbol{x}$ and $\boldsymbol{m}$ is

$$D = ||\boldsymbol{x} - \boldsymbol{m}||_A^2 = (\boldsymbol{x} - \boldsymbol{m})^T A^{-1} (\boldsymbol{x} - \boldsymbol{m}), \tag{1}$$

where d is the dimensionality of the feature vector and A is any positive definite d x d matrix [24]. The effect of A is to scale the distance along each feature axis. One of the most commonly used distance measures is the Euclidean distance, $D_E = ||\boldsymbol{x} - \boldsymbol{m}||_I$, where I is the identity matrix, and where equal weight is given to distance along each feature axis.

Partitional clustering algorithms and competitive neural networks for unsupervised learning which use the Euclidean distance metric to compute the distance between a pattern and the assigned cluster center are suitable primarily for detecting hyperspherically-shaped clusters [19]. Hence, clustering algorithms with the Euclidean distance metric have the undesirable property of splitting large or elongated (i.e., non-hyperspherical) clusters.

Other distance metrics, such as the Mahalanobis distance (MD), can also be used. The squared Mahalanobis distance between a pattern $\boldsymbol{x}$ and the i^{th} cluster center $\boldsymbol{m}_i$ is :

$$D_M = ||\boldsymbol{x} - \boldsymbol{m}_i||_C^2 = (\boldsymbol{x} - \boldsymbol{m}_i)^T C_i^{-1} (\boldsymbol{x} - \boldsymbol{m}_i), \tag{2}$$

where C_i is the covariance matrix of the samples in the i^{th} cluster [15]. Using the class-based covariance matrix normalizes distance along each axis of the d-dimensional space relative to the variance in each dimension for that class.

The use of the Mahalanobis distance in clustering has its own undesirable effects. It was noticed that its use sometimes causes a big cluster to absorb nearby smaller clusters [35], [25]. In addition to this problem, other difficulties need to be overcome before a clustering algorithm with the Mahalanobis distance can be beneficial.

1.2 Problem Statement

This thesis integrates the class-based Mahalanobis distance as a similarity measure in partitional clustering.

1.3 Summary of Current Knowledge

Even though the Mahalanobis distance is well known in the pattern recognition literature, the idea of measuring the Mahalanobis distance to each cluster based on its covariance matrix and using these measurements as a basis for partitioning is not common.

Patrick and Shen [27] suggested an interactive approach for hyperellipsoidal clustering. They used the problem knowledge to provide the computer with the initial center of the hyperellipsoidal clusters and the covariance matrix for the data in that cluster in addition to the "expert's" confidence level in those values. Their approach sequentially detects clusters, so there is no competition between different clusters, and the output clusters are affected by the order in which a cluster center is defined. This approach requires a supervisory knowledge of the data that rarely occurs, especially when visualization of the data is difficult or impossible.

In 1991, Jolion *et al.* [20] proposed an iterative clustering algorithm based on the minimum volume ellipsoid (MVE) robust estimator. At each iteration in this algorithm, the best fitting hyperellipsoidal-shaped cluster is detected by comparing the distribution of the pattern inside a minimum volume hyper-ellipsoid to a cluster generated from a Gaussian distribution. The authors realized the computational burden of finding the best-fitting hyperellipsoid in the entire feature space, so they employed a random subsampling technique and performed the clustering based on these few samples. Their approach also detects clusters sequentially which may result in a nonoptimal clustering.

In 1996, Mao and Jain [25] proposed a neural network for hyperellipsoidal clustering (HEC), which uses the following regularized squared Mahalanobis distance:

$$D_{RM}(\boldsymbol{x}, \boldsymbol{m}_i) = (1-\lambda) D_{M\epsilon}(\boldsymbol{x}, \boldsymbol{m}_i) + \lambda D_E(\boldsymbol{x}, \boldsymbol{m}_i), \qquad (3)$$

where D_E is the Euclidean distance and $D_{M\epsilon}$ is the Mahalanobis distance when a small number ϵ is added to the diagonal elements of the class covariance matrix C_i. For $0 < \lambda < 1$, D_{RM} is a linear combination of the squared Mahalanobis distance and the squared Euclidean distance. Introducing the regularized Mahalanobis distance results in a tradeoff between the previously mentioned complementary properties associated with using either the Euclidean distance or the Mahalanobis distance by itself.

This thesis analyzes the HEC algorithm and introduces another clustering method based on the Mahalanobis distance to each cluster, weighted by a factor that depends on the population of that cluster [35].

1.4 Scope

In this thesis, a new clustering technique based on the class-based Mahalanobis distance is implemented and compared with the Mahalanobis distance based HEC algorithm. Both hard and fuzzy algorithms using Euclidean distance are used as a baseline for comparison. Problems that face the use of Mahalanobis distance in clustering are analyzed and solved. The data are two-dimensional Gaussian and the well-known Iris data. Vector quantization coding of images is also considered as a practical application.

1.5 Approach

The approach taken in this investigation is composed of three steps. The first step is to implement both the LBG technique, described by Linde *et al.* [24], and the FCM, as outlined by Bezdek [7]. These techniques are validated with similar test problems to those shown in the references. The second step of the approach is to implement HEC as a neural network technique for hyper-ellipsoidal clustering using regularized Mahalanobis distance. This technique will also be tested with the same test problems as the LBG and the FCM techniques. Finally, the weighted Mahalanobis distance (WMD) algorithm is implemented and compared with the previous techniques.

1.6 Research Objectives

The research objectives for this thesis are as follows:

1. Implement the LBG and the FCM approaches and verify their performance.
2. Build a neural network algorithm for performing principal component analysis (PCA) necessary for implementing the HEC algorithm.
3. Implement a neural network-based HEC algorithm and compare its performance with the LBG and the FCM approaches.
4. Build a better Mahalanobis distance based clustering algorithm that is capable of solving the problems associated with using the Mahalanobis distance in clustering.
5. investigate the theoretical justification for the proposed method.
6. Develop sufficient understanding of how each method works to be able to draw appropriate inferences from the results, beyond simply the output partitioning.

1.7 Thesis Organization

The remainder of this thesis is organized as follows: Chapter *II* draws upon a detailed review of pertinent theory and methods relating to clustering to develop four techniques for clustering, LBG, FCM, HEC, and WMD. Chapter *III* describes the implementation of these techniques. Chapter *IV* presents and discusses the results obtained by applying these techniques to several real and synthetic data sets. Finally, Chapter *V* summarizes the results obtained in this investigation and provides recommendations for further research.

1.8 Summary

It is difficult for some problems to obtain labeled data, such as in the case of speech recognition and speaker identification. Hence, finding natural concentrations in data without supervision is necessary. Since real-life data are not isotropically distributed around well-separated centers, a system that can detect hyperellipsoidal shaped clusters is needed.

II. Theory

2.1 Introduction

This chapter focuses on the theoretical concepts necessary to understand the methods developed and used for clustering. In the sections that follow, these areas are treated:

- Background of statistical pattern classification.
- Benefits and description of clustering.
- General k-means algorithm and its hard and fuzzy variations.
- Principal component analysis
- Clustering based on the Mahalanobis distance to allow for hyperellipsoidally shaped clusters.
- Theoretical foundation of the weighted Mahalanobis distance clustering algorithm.

2.2 Pattern Classification

In statistical pattern classification, the goal is to assign a d-dimensional input pattern $\boldsymbol{x}$ to one of c classes, $\omega_1, \omega_2, \ldots, \omega_c$. If the *class conditional densities*, $p(\boldsymbol{x}|\omega_i)$, and the *a priori* class probabilities, $P(\omega_i), i = 1, 2, \ldots, c$; are completely known, the optimum strategy for making the assignment is the Bayes decision rule [12]: Assign a pattern $\boldsymbol{x}$ into class ω_j if

$$P(\omega_j|\boldsymbol{x}) > P(w_i|\boldsymbol{x}), \qquad i = 1, 2, \ldots, c; \qquad i \neq j$$

where

$$P(\omega_j|\boldsymbol{x}) = \frac{P(\omega_j)p(\boldsymbol{x}|\omega_j)}{p(\boldsymbol{x})}$$

is called the *a posteriori* probability of class ω_j, given the pattern vector $\boldsymbol{x}$ and $p(\boldsymbol{x})$ is the probability distribution function (PDF) of the measurements. This rule minimizes the overall probability of error.

Bayes rule is a special case of a more general decision rule that can be formulated as a set of c discriminant functions, $g_i(\boldsymbol{x}), i = 1,\ 2,\ \ldots\ , c$. We assign $\boldsymbol{x}$ to class ω_j if

$g_j(x) > g_i(x)$, for all $i \neq j$. The output nodes of feedforward networks, such as multilayer perceptrons and radial basis function networks, can be used to compute discriminant functions and therefore can be used as classifiers [32].

As implied before, the best discriminant functions are the a posteriori probabilities. However, in most practical applications neither the a priori probability nor the conditional PDF's for all classes are known, and only a finite amount of data is available from each class. This means that the class conditional density functions need to be estimated using just a finite number of data points. Given a collection of labeled samples, two approaches may be used to estimate the a posteriori probability of each class. The first is to assume values for the a priori probabilities, then estimate the probability distribution functions at each point. Alternatively, the a posteriori probabilities may be estimated directly from the labeled samples [12].

One class of procedures for estimating density functions entails assuming a functional form for each PDF, and then estimating parameters to fit that form to the data. One of the estimation techniques is the maximum likelihood estimation (MLE) [12]. In practice, however, it may be unjustifiable to assume the correct form for each PDF. In addition, many practical problems involve multimodal densities, while nearly all of the classical parametric densities are unimodal. The alternative to this class of procedures is to estimate the density functions directly from the data, without assuming a functional form. Because they do not involve estimates of functional parameters, these procedures are generally referred to as non-parametric techniques [12]. Examples of the well-known non-parametric density estimation techniques are k-nearest neighbor (k-NN) and Parzen window estimators. The estimated class conditional density functions can be used to calculate the discriminant functions. Neural networks also take this approach. It was shown that a multilayer perceptron (MLP) approximates the a posteriori probabilities of the classes being trained [31].

In many practical cases the data are of limited sample size and also unlabeled. This is because the class memberships are not available or because collection and labeling of a large set of sample patterns can be very costly and time consuming. For instance, recorded speech is virtually free, but labeling the speech, marking what word or phoneme is being uttered at each instant, is very expensive and time consuming [12]. In this case classification can be accomplished by finding groups in the data and then labeling these categories as specific classes by supervision.

Clustering can also be used to save time by designing a crude classifier on a small, labeled set and then modifying the design by allowing it to run without supervision on a large, unlabeled set. In addition, clustering can give the analyst insight into the nature or structure of the data especially when no prior knowledge is assumed or the data cannot be visualized. Finally, another important application of clustering is feature selection. It is important to eliminate redundant features to reduce the dimensionality of the problem. Taking into account those distinct and well separated features saves time and makes parameter estimation more accurate. For all of the above reasons, clustering is a very important tool in pattern recognition and shall be defined in the next section.

2.3 The Clustering Problem

Cluster analysis is one of the basic tools for identifying structure in data. Clustering usually implies partitioning of a collection of objects (tanks, stock market reports, images, cancerous areas in a mammogram) into c disjoint subsets. That is, to partition a set $\mathcal{H}$ of n samples $X = \{x_1, x_2, .., x_n\} \subset \mathcal{R}^d$ into subsets $\mathcal{H}_1, .., \mathcal{H}_c$. Each subset is to represent a cluster, with objects in the same cluster being somehow "more similar" than samples in different clusters. In other words, objects in a cluster should have common properties which distinguish them from the members of the other clusters.

In fuzzy clustering, the clusters are not subsets of the collection but are instead fuzzy subsets as introduced by Zadeh [36]. That is, the "clusters" are functions assigning to each object a number between zero and one which is called the membership of the object in the cluster. Objects which are similar to each other are identified by the fact that they have high memberships in the same cluster. It is also assumed that the memberships are chosen so that their sum for each object is one. So, the fuzzy clustering is, in this sense, also a partition of the set of objects.

Each cluster may be represented by a reproduction vector, $\boldsymbol{m}_i$, that is usually chosen to be the Euclidean center of gravity, or the centroid of the samples in that cluster. Hence, it is called the cluster center.

To measure a clustering algorithm performance, one defines a criterion function that measures the clustering quality of any partition of the data. The goal is to find the partition that minimizes the criterion function. The approach most frequently used in seeking optimal partitions is iterative optimization [12]. The basic idea is to start with an initial partition and iteratively change the location of cluster centers and reassign data points to the different clusters in a way that maximizes the criterion function.

Clustering or unsupervised learning algorithms can be grouped into two categories: hierarchical and partitional. A hierarchical clustering is a nested sequence of partitions, whereas a partitional clustering is a single partition [18]. Commonly used partitional clustering algorithms are the k-means algorithm and its variations (e.g., ISODATA [2]) which incorporates some heuristics for merging and splitting clusters and for handling outliers. Competitive neural networks are also used to cluster input data. The well-known examples are Kohonen's Learning Vector Quantization (LVQ) and self-organizing map [23], and Adaptive Resonance Theory (ART) models [8].

A clustering algorithm in this context is equivalent to a vector quantizer (VQ) where the object is to find a system for mapping a sequence of continuous or discrete vectors into a digital sequence suitable for communication or storage in a digital channel [17]. The goal of vector quantizers is data compression. They are used to reduce the bit rate while maintaining the necessary fidelity of the data [17]. When used in a vector quantization context, cluster centers are termed codewords and the set of all codewords is called a codebook. To define the performance of a quantizer in producing the "best" possible reproduction sequence requires the idea of a distortion measure. Despite the differences between clustering algorithms and vector quantizers, their ultimate goal of exploiting the similarity between the samples allows us to use the terms clustering and vector quantization interchangeably throughout this thesis.

Most variations in the design of a vector quantizer or a clustering algorithm are in the choice of a quantitative measure of similarity and in the definition of the criterion function, or distortion. Hence, these two critical topics are the subject of the following subsections.

2.3.1 Similarity Measures. The first important question in a clustering algorithm design is how to determine "more similar". Similarity measures can be in a form of non-metric similarity functions or distance metrics. Duda and Hart [12] list a number of functions $s(\boldsymbol{x}, \boldsymbol{m})$ whose values are higher if the vectors $\boldsymbol{x}$ and $\boldsymbol{m}$ are somehow similar. An example measure is the normalized inner product which is the cosine of the angle between the two vectors. Other similarity functions include the two-sided tangent distance and Tanimoto distance which is especially useful if the features are binary [12].

Distance measures are more often used to measure similarity. The smaller the distance between two patterns, the greater the similarity. The assumption made in using these measures is that similarity between objects is measured by the distance between their corresponding data vectors. So, if the distances to the mean of a cluster are small, the points in the cluster are also close to each other and therefore assumed to be "similar".

Mao and Jain [25] suggest that Euclidean distance is the most commonly used distance metric in practice. However, choosing the Euclidean distance to measure similarity implies an isotropic feature space weighting. Consequently, a clustering algorithm with Euclidean distance favors hyperspherically shaped clusters of equal size. The clusters in a data set are not always compact and isolated in the sense that the between-cluster variation is much higher than the within-cluster variation. Therefore, using the Euclidean distance has the undesirable effect of splitting large as well as elongated clusters under some circumstances [25]. It is easy to find examples of data sets in real applications which do not have spherically shaped clusters of equal size. For example, Fig. 1 shows the two-dimensional projection of the well-known Iris data onto a plane spanned by the first two principal eigenvectors. Since we know the category information of the patterns in the Iris data set, we label each pattern by its category in the projection map. There are two obvious clusters, but neither of them has a spherical shape. From Fig. 1, we can see that

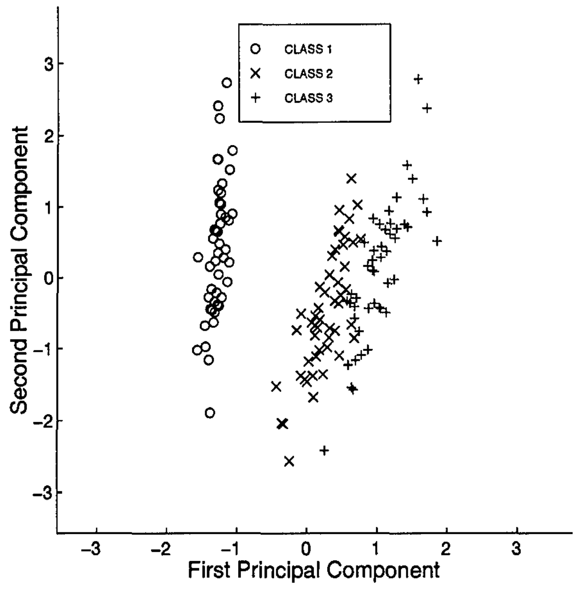


Figure 1. Projection of the Iris data set onto the plane spanned by the first two principal components

the larger cluster is a mixture of two classes (Iris Versicolor and Iris Virginica) which have nonspherical distributions.

An alternative distance metric that depends on the distribution of the data itself is the Mahalanobis distance defined in Eqn. (2) and rewritten here for convenience.

$$D_M = ||\boldsymbol{x} - \boldsymbol{m}||_C^2 = (\boldsymbol{x} - \boldsymbol{m})^T C^{-1} (\boldsymbol{x} - \boldsymbol{m}) \tag{4}$$

where C is the covariance matrix of a pattern population, $\boldsymbol{m}$ is the mean vector, and $\boldsymbol{x}$ represents a variable pattern. The Mahalanobis distance is especially useful if the statistical properties of the population are to be considered.

Figure 2 shows a scenario of two-cluster solutions by vector quantization using the Euclidean and Mahalanobis distance measures. It is clear from the figure that using the Euclidean distance results in undesired grouping of the two classes. On the other hand, using the Mahalanobis distance allows the natural grouping of the two clusters.

The equi-Euclidean distance surface defines a hyper-sphere of unity aspect ratio, which is the ratio between the major axis to the minor axis in an ellipse. On the other hand, the equi-Mahalanobis distance surface defines a hyperellipsoid of any shape which

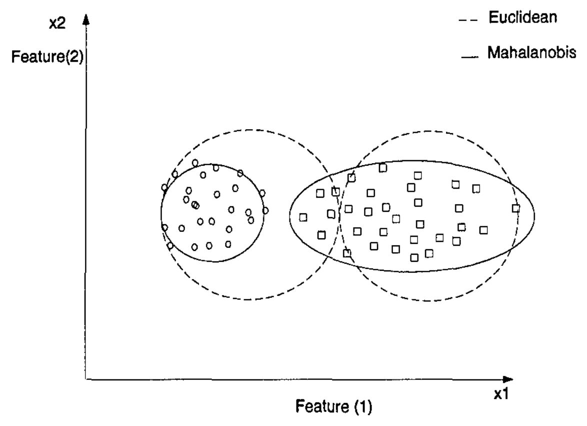


Figure 2. Results of clustering the shown patterns into two classes using the Euclidean and the Mahalanobis distance metrics

implies different orientations of the major axis and any aspect ratio. This shows that the modeling capabilities of the Mahalanobis distance-based clustering is much greater than that of the Euclidean distance-based algorithms, especially as the dimensionality increases wherein the probability of having hyperspherically shaped cluster is virtually zero.

It is clear from Figure 2 that if we could describe the shape of each cluster by its centroid and the sample covariance matrix of the data points assigned to that cluster, then the most useful similarity measure would be the Mahalanobis distance.

2.3.2 Optimality Criterion. Choice of optimality criterion is a very important issue in the design of a clustering algorithm. Especially in higher dimensions, one cannot visually determine how good the resulting clusters are. One approach is to check with a criterion function. The criteria, or performance indices, are often specified as a function of the memberships whose minima or maxima define "good" clustering. The algorithm then becomes a numerical procedure for finding memberships which optimize the objective function.

Let $D(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)$ be the distortion between a pattern $\boldsymbol{x}_i$ and the closest cluster center $\hat{\boldsymbol{x}}_i$. Or equivalently, this can be thought of as the cost of reproducing any input vector $\boldsymbol{x}_i$ as a reproduction vector $\hat{\boldsymbol{x}}_i$. The performance of the system can be quantized by an average distortion $E\{D(X, \hat{X})\}$ between the input and the final reproduction. Assuming that the process is stationary and ergodic, then the mathematical expectation is, with probability one, equal to the long term sample average [17]

$$E\{D(X,\hat{X})\} = \lim_{n\to\infty} \frac{1}{n} \sum_{j=0}^{n-1} D(\boldsymbol{x}_j, \hat{\boldsymbol{x}}_j) \tag{5}$$

Equation (5) implies that if the input source statistics do not change with time and if the time average equals the ensemble average then the resulting sample average distortion should be nearly the expected value, and the same codebook used on future data should yield approximately the same averages.

As for the choice of distortion measure, it ought to to be tractable, computable, and subjectively meaningful so that large or small quantitative distortion measures correlate with bad and good subjective quality [17].

The simplest and most widely used distortion measure, although lacking the above properties, is the squared error distortion measure,

$$D(\boldsymbol{x},\hat{\boldsymbol{x}}) = ||\boldsymbol{x} - \hat{\boldsymbol{x}}||_I^2 = \sum_{i=0}^{d-1}(x_i - \hat{x}_i)^2. \tag{6}$$

Substituting Eqn. (6) in Eqn. (5) yields the mean squared error (MSE) criterion function. Dunn [13] developed the first fuzzy extension to the least mean square error approach to clustering and this was generalized by Bezdek [6] to a family of algorithms. Let n be the total number of samples, n_i the number of samples in $\mathcal{H}_i$, and $\boldsymbol{m}_i$ be the mean of those samples. The generalized mean squared error is then defined by:

$$J_{GMSE} = \frac{1}{n} \sum_{i=1}^{c} \sum_{j=1}^{n_i} (u_{ij})^m ||\boldsymbol{x}_j - \boldsymbol{v}_i||_A^2 \tag{7}$$

where u_{ij} is the membership of the i^{th} pattern to the j^{th} cluster, m is a weighting exponent strictly greater than one, and A is any positive definite matrix [7].

Many algorithms were designed based on minimizing J_{GMSE}. In these algoritms, the J_{GMSE} is monotonically nonincreasing function as the number of iterations increases.

Modifying the criterion function to allow the use of a generalized distance as a distance metric is possible by defining A. Linde *et al.* [24] and Bezdek [7] generalized A to any d x d positive definite matrix. They showed that if the covariance matrix for all the data is used, the Mahalanobis distance is the basis for the partitioning.

Clustering using the global covariance matrix results in a monotonically decreasing objective function. However, all the clusters may not have the same distribution (shape) as that of the whole data set. Hence, using the global covariance matrix for each cluster may not give the best grouping of the data.

Other common criterion functions are the average squared distances between samples in a cluster domain and the average squared distances between samples in different cluster domains [33].

Backer and Jain [1] suggested a goal-directed approach to the cluster validity problem. The goal can be minimizing the classification error rate. Alternatively, the clustering performance can be considered good if the clusters were dense, which means minimum volume of the equi-Mahalanobis distance ellipsoids, or the clusters were far apart. Gath and Geva [14] also considered a large number of the data points in the vicinity of the cluster center as an indication of good clustering. In Section 2.6.4 we will see that the goal is maximizing the likelihood function.

2.4 The Batch k-means Algorithm

A well known algorithm for the design of a locally optimal codebook with iterative codebook improvement is the generalized Lloyd algorithm (GLA) [15]. The two steps in each iteration of this algorithm are:

Step 1: Given a codebook $M = \{\boldsymbol{m}_i \ ; i = 1, \ldots, k \}$ obtained from the l^{th} iteration, assign each data point to the *closest* codeword.

Step 2: Obtain the codebook M_{l+1} by computing the centroid of each cluster based on the partitioning of Step 1.

where the closest codeword is typically found with a distance metric based on the assumption that similar feature vectors are likely to lie close to each other in the feature space. The above algorithm is usually terminated when the codewords stop moving or the difference between their locations in consecutive iterations is below a threshold.

The GLA algorithm is widely known as the batch k-means algorithm when the distortion measure used is the sum of squared distances from all points in a cluster domain to the cluster center [33]. A sequential k-means, where the update to the cluster centers occurs after each pattern being classified, is proposed by McQueen [26].

In the next sections, two methods that use variations on the described k-means algorithm are presented. We will discuss how these methods differ in calculating the membership function of the data to the reproduction vectors.

2.4.1 The LBG Algorithm. The first algorithm is described by Linde, Buzo, and Gray in 1980 [24], and is also known as the LBG algorithm for short. The LBG Algorithm is similar to k-means, but the distortion measure is general. A survey of different distortion measures is provided in the paper. It was shown that the best partition is obtained by choosing the nearest-neighbor codeword to each input, and no reproduction alphabet can yield a smaller average distortion than that containing the centroids of the subsets. The GLA iterative method with these rules is guaranteed to result in a non-increasing average distortion (based on the distance from the cluster centers to the data points within the clusters) after each iteration, and it terminates in a local minimum when the average distortion stops changing significantly.

2.4.2 The Fuzzy c-means Algorithm. In a "hard" clustering algorithm such as the k-means algorithm, each pattern is to be assigned to only one cluster. However, this "All-or-None" membership constraint is not realistic, since many pattern vectors may have characteristics of more than one class. Hence came the idea of assigning a set of membership function values for each pattern to every one of the different classes. This results in a fuzzy boundary rather than a hard one.

The first and most widely used fuzzy clustering algorithm is the fuzzy c-means algorithm [34]. It is an iterative procedure for finding a membership matrix, U, and a set of cluster centers, M, to minimize the generalized mean squared error objective function, see Eqn. (7). By minimizing J_{GMSE}, one should obtain high memberships in cluster j for data which are close to the vector $\boldsymbol{m}_j$.

The vectors $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_c$ can be interpreted as prototypes for the clusters represented by the membership functions, and are also called cluster centers. In order to minimize the objective function, the cluster centers and membership functions are chosen so that high memberships occur for points close to the corresponding cluster centers. The number m is called the exponent weight. It can be chosen to "tune out" noise in the data. The higher the value of m, the less the contribution of those data points whose memberships are uniformly low to the objective function. Consequently, such points tend to be ignored in determining the centers and membership functions [34]. The actual construction of the FCM algorithm is based on the following set of equations which are necessary conditions for $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_c$ and the membership matrix U to produce a local minimum:

$$\boldsymbol{m}_i = \frac{\sum_{j=1}^{n} (u_{ij})^m \boldsymbol{x}_j}{\sum_{j=1}^{n} (u_{ij})^m}, \tag{8}$$

and

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{(\boldsymbol{x}_j - \boldsymbol{m}_i)^T A^{-1} (\boldsymbol{x}_j - \boldsymbol{m}_i)}{(\boldsymbol{x}_j - \boldsymbol{m}_k)^T A^{-1} (\boldsymbol{x}_j - \boldsymbol{m}_k)} \right)^{\frac{2}{m-1}}}. \tag{9}$$

The update equations are obtained by differentiating the objective function with respect to the components of V and U subject to the constraint that $\sum_{i=1}^{c} u_{ij} = 1$. No closed form solution has been found for this differentiation, but these equations are the basis for an iterative procedure which converges to a local minimum for the objective function [34]. One could start with an initial guess of the value for the membership values, U, for a specific choice of m and c. Using these memberships and Eqn. (8), one computes cluster centers, then using these cluster centers and Eqn. (9) recomputes memberships and so forth, iterating back and forth between Eqn. (8) and Eqn. (9) until the memberships or cluster centers for successive iterations differ by no more than some prescribed value.

In the cases where classification is the goal, Eqn. (9) can be used as the basis where maximum membership function value u_{ij} means smaller distance and higher likelihood that the data point $\boldsymbol{x}_i$ is a member of the j^{th} cluster. The class membership function values provide a measure of similarity by which a soft decision can be made with a degree of confidence.

2.5 Principal Component Analysis

One of the topics that is used in signal processing and used in the techniques to be presented in the next sections is the principal components analysis (PCA). The goal is to find the maximum variance in the data and to transform the data into a space where the features are uncorrelated.

Let C_x be the covariance matrix of the data set X. Since C_x is real and symmetric, it is possible to find a set of orthonormal eigenvectors [16]. Let $\mathbf{e}_i$ and ψ_i, $i = 1, 2, \ldots, d$, be the eigenvectors and the eigenvalues of C_x. If we arrange the eigenvalues in decreasing order so that, $\psi_1 \geq \psi_2 \geq \ldots \geq \psi_d$, a transformation matrix whose rows are the corresponding eigenvectors of C_x is defined to be Φ.

If $\boldsymbol{m}_x$ is the mean of the samples X, the transformation

$$\boldsymbol{y} = \Phi(\boldsymbol{x} - \boldsymbol{m}_x) \tag{10}$$

is called principal component, Karhunen-Loève, or Hotelling transformation.

It can be shown that, the mean of Y is zero and its covariance matrix is given by [16]:

$$C_y = \Phi^T C_x \Phi = \Psi = \text{diag}\{\psi_1^2, \ldots, \psi_d^2\}, \tag{11}$$

which means that the features of Y are uncorrelated. Furthermore, if we divide each eigenvector by the square root of the corresponding eigenvalue, the resultant covariance matrix, C_y, will be the identity. Hence, this scaled principal component analysis is called the *whitening transform.*

Another interesting property of this transform can be seen if we let

$$y = \Psi^{-1/2}\Phi x, \tag{12}$$

and

$$v = \Psi^{-1/2}\Phi m_x, \tag{13}$$

for any vector $x \in X$. Computing the Euclidean distance between v and y yields

$$D_E(y, v) = (y - v)^T(y - v). \tag{14}$$

Substituting Eqn. (12) and Eqn. (13) and using the fact that for any orthonormal matrix

$$\Phi^T = \Phi^{-1} \tag{15}$$

it can be shown that Eqn. (14) becomes

$$D_E(y, v) = (x - m_x)^T C_x^{-1}(x - m_x), \tag{16}$$

which is the definition of the Mahalanobis distance between a vector x and a cluster center m_x. In other words, this says that the Mahalanobis distance between an input pattern x and a cluster center m_x in the input space is equivalent to the Euclidean distance between them in the transformed space, i.e.,

$$D_M(x, m_x) = D_E(y, v). \tag{17}$$

For any $q \leq d$, the projection of x onto the span of $\{e_1, e_2, \ldots, e_q\}$ accounts for the maximum fraction of the total sample variance in x that can be accounted for by a linear projection onto a q-dimensional vector subspace. Plotting the first few principal components allows us to visually examine the data that account for most of its variance. These properties of the principal component analysis will be utilized in the Mahalanobis distance based algorithms.

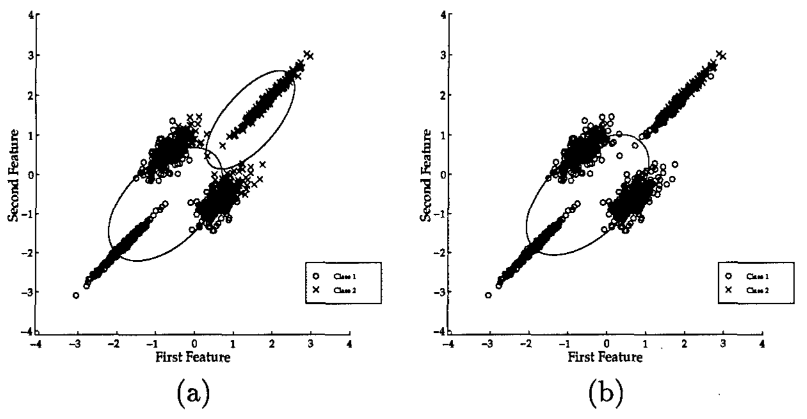


Figure 3. The case where a large cluster engulfs a neighboring smaller cluster. (a) The first cluster has more samples than the second class. (b) More samples are assigned to the cluster of the left as the iterative process continues

2.6 Clustering Based on The Mahalanobis Distance

We have seen that the LBG and the FCM algorithms allowed A to be any positive definite matrix. Any selection of A implies that the algorithm is looking for clusters that represent concentration of the data about the centers, M, whose shape are determined by the matrix A. As stated before, if $A = I$, the clusters identified tend to be spherical. In [7] and [24], the authors suggested that A can be the covariance matrix of all the samples. This is again considered imposing the shape of the overall distribution of the data set on each cluster, which might not be the case in practical situations. Assuming a different matrix for each cluster has the effect of detecting rather than imposing shapes in the data.

In the next sections, we will first describe some of the problems that faces the use of Mahalanobis distance in clustering. Next, we discuss two methods that use the Mahalanobis distance from each cluster center as a similarity measure. We will discuss how these methods try to solve the described problems.

2.6.1 Problems with Integrating The Mahalanobis Distance in Clustering. There are some difficulties associated with computing the class-based Mahalanobis distance as a similarity measure in a k-means-type algorithm:

1. The covariance matrices needed for the Mahalanobis distance calculation in the first iteration cannot be determined *a priori.*

2. If the number of patterns in a cluster is small compared to the input dimensionality d, then the d x d sample covariance matrix of the cluster may be singular.

3. The k-means clustering algorithm with the Mahalanobis distance tends to generate unusually large or unusually small clusters [35], [25]. Figure 3 shows the case of large cluster engulfing the neighboring cluster. Points on the perimeter are of equal Mahalanobis distance to the cluster centers. Since the first cluster exhibits more variations in both directions, its covariance matrix is larger. Hence the Mahalanobis distance to the cluster center with a larger A is more likely to be smaller and more samples are assigned to that cluster in subsequent iterations.

4. When performing the clustering based on the individual covariance matrices to allow for each cluster to have its distinct shape, J_{GMSE} does not result in a monotonically decreasing function as the number of iterations increases.

   Figure 4 shows that as the data cluster together in smaller groups, the variance of data within a cluster is now less which will be reflected in the covariance matrix. The distance between a point $\boldsymbol{x}$ and the mean $\boldsymbol{m}$ is normalized by the smaller variance. So while the Euclidean distance is still the same, the Mahalanobis distance got larger in the second iteration. The change in the Mahalanobis distance is indicated by the equi-Mahalanobis distance ellipses. Therefore, the sum of the Mahalanobis distances increases.

2.6.2 The Hyperellipsoidal Clustering (HEC) Neural Network. As described in the previous sections, the LBG or the FCM algorithm performs clustering using either the Euclidean distance or using the Mahalanobis distance based on the global covariance matrix. Mao and Jain [25] proposed a neural network which implements partitional clustering using the following regularized Mahalanobis distance between the data point $\boldsymbol{x}_i$ and the cluster center $\boldsymbol{m}_k$:

$$D_{RM}(\boldsymbol{x}_i, \boldsymbol{m}_k) = (\boldsymbol{x}_i - \boldsymbol{m}_k)^T[(1-\lambda)(C_k + \epsilon I)^{-1} + \lambda I](\boldsymbol{x}_i - \boldsymbol{m}_k) \tag{18}$$

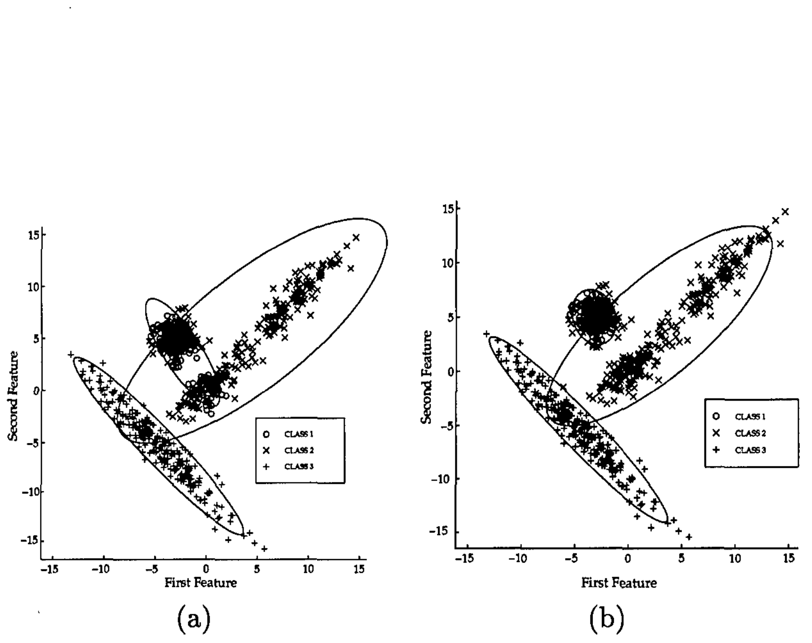


Figure 4. The results of clustering using the Mahalanobis distance in subsequent iterations. J_{GMSE} is smaller in the first iteration (a) than in the next iteration (b).

where λ and ϵ are called the regularization parameters. Note that this regularized Mahalanobis distance takes the shape of each cluster into consideration. When λ=0 and ϵ=0, D_{RM} becomes the squared Mahalanobis distance. When λ= 1, D_{RM} reduces to the squared Euclidean distance. When $0 \leq \lambda \leq 1, D_{RM}$ is a linear combination of the squared Mahalanobis distance and the squared Euclidean distance. Therefore, λ can be used as a parameter to control the degree that the distance measure deviates from the commonly used squared Euclidean distance. On the other hand,ϵ is added to prevent the singularity of the covariance matrix C_k as shall be discussed later.

Equation (17) indicates that the computation of the Mahalanobis distance can be decomposed into two steps; first, projecting the input pattern into the whitened space and second, computing the Euclidean distance in the whitened space.

Neural networks provide a suitable architecture for implementation of many signal processing techniques. hence, the authors implemented the two-step computation using a two-layer network whose first and second layers compute the whitening transform and the Euclidean distance, respectively. Then, the regularized distance can be computed by the weighted sum of two Euclidean distances: one in the original input space and one in the whitened space.

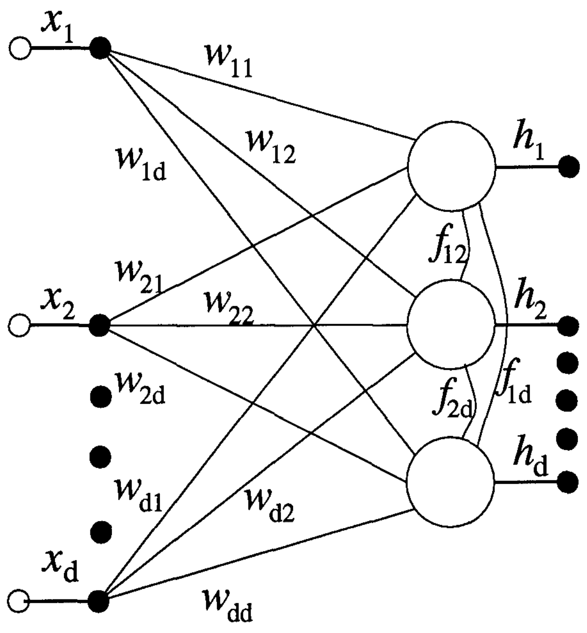


Figure 5. Principal component analysis neural network

Since it is necessary to find the eigenvectors and eigenvalues before such a regularized Mahalanobis distance can be calculated, a neural network version of principal component analysis algorithm was chosen to perform the whitening. In the next sections, a description of the PCA neural network algorithm is outlined. Then, the architecture for the HEC network is described. Finally, a discussion of the HEC algorithm is provided.

2.6.2.1 Principal Component Analysis Subnetworks. Rubner and Tavan [30] presented a two-layered network whose weights converge to the eigenvectors of the covariance matrix of the input patterns, see Fig. 5.

All the output nodes are hierarchically organized such that the output node i is connected to the output node j with connection strength f_{ij} if and only if $i < j$.

For an input pattern $\boldsymbol{x}$, the PCA net outputs h_j

$$h_j = \boldsymbol{w}_j \cdot (\boldsymbol{x} - \boldsymbol{m}_x) + \sum_{l<j} f_{lj} h_l. \tag{19}$$

The weights on the connections between the input nodes and the output nodes of the PCA subnetwork are adjusted according to the Hebbian rule, i.e.,

$$\Delta w_{ij} = \gamma(\boldsymbol{x}_i - \boldsymbol{m}_i)h_j, \qquad i,j = 1,2,\ldots,d; \tag{20}$$

where γ is the learning rate. The lateral weights are updated according to the anti-Hebbian rule

$$\Delta f_{ij} = -\mu h_l h_j, \qquad l,j = 1,2,\ldots,d, \qquad l < j; \tag{21}$$

where μ is another positive learning rate. Rubner and Tavan demonstrated that following the above update equations forces the lateral weights to vanish and the activities of the output cells to become uncorrelated. Then the weights on the connections to the j^{th} output layer, $\boldsymbol{w}_j$, are the eigenvector corresponding to the j^{th} largest eigenvalue. The j^{th} eigenvalue is estimated by the variance of h_j after the network converges.

2.6.2.2 Architecture for Hyperellipsoidal Clustering Network. Figure 6 shows the diagram of the network architecture of HEC. The input layer has d input nodes corresponding to d features. The hidden layer is composed of k x d linear nodes which are grouped in k subnetworks with d nodes each, where k is the number of clusters. Each network is designed to perform the whitening transform of a cluster. The output layer has k nodes corresponding to k clusters, and is basically a "winner-take-all" network. The lateral connections have not been shown. Competitive learning is performed in the output layer. The input layer is fully connected to the output layer. All the output nodes in the k^{th} subnetwork, however, are connected only to the k_{th} node in the output layer.

Since each cluster needs its own whitening transform, the weights in the PCA subnetworks are updated only when these patterns in the corresponding cluster are presented. Moreover, in the PCA learning mode, the centroid of the corresponding cluster should be subtracted from the input pattern vectors. Therefore, before the PCA learning is invoked, the cluster centroid must be learned.

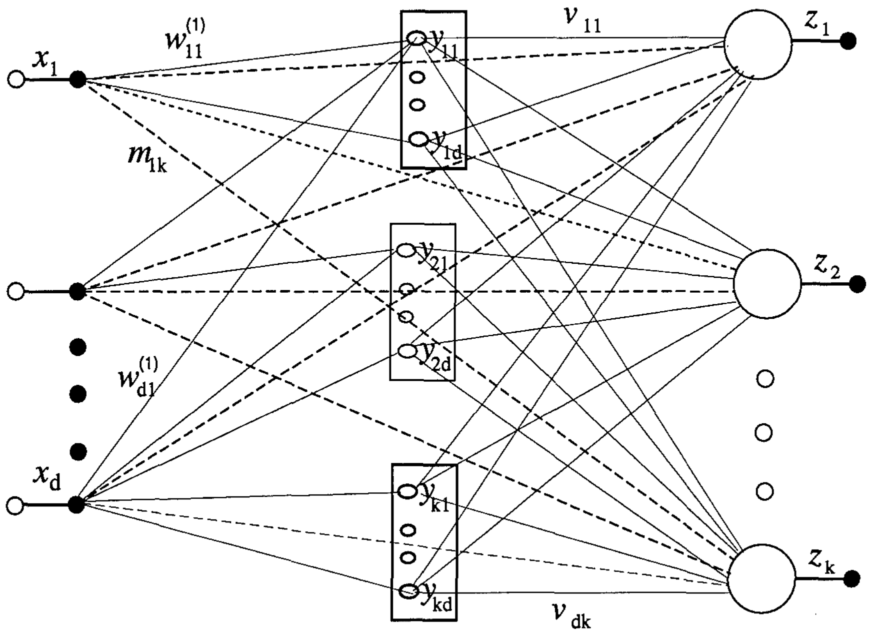


Figure 6. Architecture of the neural network for Hyperellipsoidal clustering (HEC)

Each node in the output layer computes the weighted squared Euclidean distance between its inputs (both the input pattern and outputs of the connected subnetworks) and the stored pattern on the connections.

2.6.2.3 The HEC Algorithm. The algorithm starts with initial cluster centers, eigenvectors, and eigenvalues and then iteratively measures the regularized Mahalanobis distance to each cluster center. The input pattern then updates the PCA subnetwork for the class that yields the minimum D_{RM}. The algorithm updates the cluster centroids using a batch vector quantization algorithm. Afterwards, a new partition of the data set into k clusters is formed. Then, the HEC algorithm reinvokes the PCA learning to update the individual whitening transforms (PCA subnetworks) for all the k clusters. If the stopping criterion is not met, the value of the regularization parameter λ is reduced. The algorithm is terminated either when the cluster membership of input patterns does not change or the maximum number of learning cycles is reached.

The HEC algorithm tries to solve the problems associated with using the Mahalanobis distance in clustering, (see Section 2.6.1). It starts with preference for the Euclidean distance to address the first problem concerning initial values for the individual covariance matrices. After the data have been partitioned using a batch vector quantization algorithm with Euclidean distance, the means and the covariance matrices of the individual clusters are calculated and used for upcoming iterations.

The second problem of dealing with ill-posed conditions is solved by the introduction of the regularization parameter ϵ. The role of ϵ is to convert a singular covariance matrix to a nonsingular matrix by adding a diagonal matrix with small elements. This actually forces the cluster to have a non-zero extent in all dimensions.

The authors suggested the use of larger value of λ when the covariance matrix C cannot be reliably estimated or learned. In addition, this regularized MD was hoped to solve the problem of large clusters swallowing smaller nearby ones. However, the algorithm starts with initial value of $\lambda = 1$ and decreases with each iteration. This suggests that this problem is more likely to occur very often only during the first few cycles of learning.

In this algorithm the performance was not evaluated by minimization of the cost function defined in the paper as the J_{GMSE} with $m = 1$. However, the quality of the resultant clustering is assumed to correlate with the compactness of the output clusters. The compactness of a cluster is measured by the significance level of the Kolmogorov-Smirnov test on the distribution of the Mahalanobis distances of patterns in a cluster to the cluster center under the Gaussian cluster assumption. Moreover, it was not shown how this is incorporated in the algorithm

2.6.3 The Weighted Mahalanobis Distance Clustering Algorithm. The idea behind the proposed method is to allow each cluster to have its own shape implied by the covariance matrix of the samples within that cluster. A cluster would attract those data points which enhance its shape and reject those which do not fit. In other words, each vector x will be assigned to the cluster to which it best integrates regardless of the Euclidean separation between that data point and that cluster's centroid.

In this algorithm, we modify the GLA described in Section 2.4 such that in each iteration we assign a pattern $\boldsymbol{x}$ to the cluster that yields the minimum *weighted* Mahalanobis distance, $D_i = w_i * ||\boldsymbol{x} - \boldsymbol{m}_i||^2_{C_i}$, where w_i is the cluster weight and its computations will be discussed later. In the second step, we update $\boldsymbol{m}_i$ and C_i by computing the mean and the covariance matrix of the data points of each cluster based on the partitioning of the first step. The algorithm is terminated when the codewords stop moving.

Two ways to get an initial codebook are discussed here. The first method treats the whole data set as a unique cluster and grows the codebook to the desired number of clusters, k, by *splitting*. In splitting, small perturbation vectors $\boldsymbol{\rho}$ are added to every codeword. We made $\boldsymbol{\rho}_i$ for the i^{th} cluster to be in the direction of the eigenvector corresponding to the maximum eigenvalue of A_i. This provides a systematic approach of distributing the codewords since each cluster center splits in a particular direction based on the cluster's shape. The second approach to get an initial codebook is used when the desired codebook size is not an integer power of two. This method uses the Karhunen-Loève transformation to place the initial codewords along the principal component axes of the data's covariance matrix. For the first iteration, we use the global covariance matrix as A_i for all the codewords.

If the number of data points in any cluster is less than the dimensionality of the data, then A_i might be singular which prevents computing the inverse. Thus, we add a matrix with small diagonal elements to the covariance matrix to prevent the singularity.

The introduction of the weight w is due to the fact that the use of Mahalanobis distance alone in clustering sometimes causes a large cluster to attract members of neighboring clusters. This leads to unusually large and unusually small clusters. We evaluate w_i as the ratio of the number of samples in $\mathcal{H}_i$ to the size of the training set. This approach is appealing as it does not rely on an arbitrary constant or user adjustable parameter. Even though this weight penalizes large clusters, it does not prevent a cluster from having more samples if they are much closer to it than other codewords. This is similar to *conscience* in neural networks which rewards a balanced representation of the data points [10], [29].

In trying to deal with the problem of nondecreasing J_{GMSE}, one can suggest the use of another goal-oriented criterion function such as the sum of the volume of the ellipsoids. Nevertheless, we recall that the update equations for the LBG and the FCM algorithms were computed based on minimizing J_{GMSE} when A was fixed for all iterations and clusters.

However, as the data cluster together in smaller groups, the variances decrease and the Mahalanobis distance increases as was explained before. Looking at the J_{GMSE} criterion function again and allowing A_i to be variable

$$J_{GMSE} = \frac{1}{n} \sum_{i=1}^{c} \sum_{j=1}^{n_i} (u_{ij})^m (x - m)^T A_i^{-1} (x - m) \tag{22}$$

it is clear that there is a need to restrict A_i somehow in order to obtain a nontrivial solution. Otherwise, the minimum of J_{GMSE} would be given by A_i^{-1}=0, which corresponds to a huge cluster with infinite variation in all directions.

One way to solve this problem is to force the determinant of all clusters to have a unity value. In d-dimensional space, a surface of constant Mahalanobis distance D_M is a hyperellipsoid. The volume of this d-dimensional hyperellipsoid is [12:24]

$$V = V_d |C|^{1/2} D_M^d, \tag{23}$$

where V_d is the volume of an d-dimensional unit radius hypersphere, which only depends on the dimensionality of the space:

$$V_d = \frac{\pi^{d/2}}{(d/2)!}, \quad d \text{ even}, \tag{24}$$

$$V_d = \frac{2^d \pi^{(d-1)/2} \left(\frac{d-1}{2}\right)!}{d!}, \quad d \text{ odd}. \tag{25}$$

Hence, by restricting $|C| = 1$, constant Mahalanobis distance surfaces now enclose constant volumes.

This choice of this constraint has been criticized because it seemed somewhat arbitrary [34]. However, it has the effect of normalizing the volume enclosed by the equi-Mahalanobis distance hyperellipsoid to a constant volume for all clusters while maintaining the distinct shape of each cluster.

After the normalization of the determinants, the J_{GMSE} distortion has become monotonically decreasing as a function of clustering iterations. This means that now we can use the criterion function in Mahalanobis distance based algorithms to find the number of clusters k or to define a stopping criterion. In the next section, we will give an analysis of the weighted Mahalanobis distance algorithm as an approximation to the famous expectation maximization method as applied to the problem of maximum likelihood estimation in the Gaussian mixture model.

2.6.4 Relation to Maximum Likelihood Estimate in The Gaussian Mixture Model (GMM). The use of different covariance matrices for different clusters was a motivation to investigate the similarity between the WMD algorithm and the Gaussian mixture model. This presentation of GMM will follow the outline of Reynolds [28], to which the reader is referred for a more extensive treatment of the subject.

A Gaussian mixture density is a weighted sum of L component densities as given by the equation:

$$p(x|\Theta) = \sum_{i=1}^{L} p_i b_i(\boldsymbol{x}), \tag{26}$$

where $\boldsymbol{x}$ is a d-dimensional random vector, $b_i(\boldsymbol{x})$ are the component densities and p_i are the mixture weights for $i = 1, \ldots, L$. Each component density is a d-variate Gaussian function of the form

$$b_i(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|C_i|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{m}_i)^T C_i^{-1}(\boldsymbol{x}_t - \boldsymbol{m}_i)\right], \tag{27}$$

where $\boldsymbol{m}_i$ represents the mean vector and C_i is the covariance matrix [12:23].

The mixture weights satisfy the constraint that $\sum_{i=1}^{L} p_i = 1$, which ensures the mixture is a true probability density function. The complete Gaussian mixture density is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities which is represented by the notation

$$\Theta = (p_i, \boldsymbol{m}_i, C_i), \qquad i = 1, \ldots, L. \tag{28}$$

It is known that sometimes the observation density is not well represented by any one parametric PDF, such as a single Gaussian or Laplacian density. Hence, the GMM can be viewed as a parametric PDF based on a linear combination of Gaussian basis functions capable of representing a large class of arbitrary densities. One of the powerful attributes of the GMM is its ability to form smooth approximations to arbitrarily shaped densities. Whereas the classical uni-modal Gaussian model could model a feature distribution only by a position (mean vector) and an elliptic shape (covariance matrix), the GMM has L mean vectors, covariance matrices and weight parameters which allow it much greater modeling capabilities.

Maximum likelihood parameter estimation is a general and powerful method for estimating the parameters of a stochastic process from a set of observed samples. It is based on finding the model parameters, Θ, which are the most likely to have produced the set of observed samples. This is accomplished by finding the parameters which maximize the model's likelihood function over a set of observations. For a set of observations referred to collectively as $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$, the likelihood function for a model Θ is defined as the joint probability density of X treated as a function of Θ. Thus, the probability density $p(X|\Theta)$ is called a *likelihood function* when it is considered as a function of Θ. This likelihood function can be calculated as:

$$p(X|\Theta) = \sum_{I} p(X, I|\Theta), \tag{29}$$

where

$$p(X, I|\Theta) = \prod_{t=1}^{n} p_{it} b_{it}(\boldsymbol{x}_t), \tag{30}$$

is the joint PDF of X and I, and the notation $\sum_I$ denotes the sum over all possible assignments of the data set.

A necessary condition for the maximum likelihood parameter estimates is that they satisfy the likelihood equation

$$\frac{\partial p(X|\Theta)}{\partial \Theta} = 0. \tag{31}$$

Unfortunately, attempting to solve Eqn. (31) directly for the GMM parameters does not yield a closed form solution [28]. However, an iterative parameter estimation procedure which is a special case of the Expectation Maximization (EM) algorithm can be implemented to find the Maximum likelihood estimates [3]. The EM algorithm is the basis for several parameter estimation procedures in the area of statistical data analysis. The widespread use of the EM algorithm stems from the fact that it guarantees a non-decreasing likelihood function after each iteration. The basic idea of the EM algorithm is, beginning with an initial model, Θ, estimate a new model, $\hat{\Theta}$, such that $p(X|\hat{\Theta}) \geq p(X|\Theta)$. The new model then becomes the current model and the process is repeated until some convergence threshold is reached.

Reynolds [28] stated that Baum [3] has derived an auxiliary function,

$$Q(\Theta, \hat{\Theta}) = \sum_{I} p(X, I|\Theta) \log p(X, I|\hat{\Theta}), \tag{32}$$

which exhibits the property that finding $\hat{\Theta}$ that maximizes $Q(\Theta, \hat{\Theta})$, results in an increase in the observed data likelihood, which is the goal. Reynolds [28] showed that the estimation equations for the model's parameters are as follows:
The mixture weights are obtained by:

$$p_i = \frac{1}{n} \sum_{t=1}^{n} P(w_i|\boldsymbol{x}_t, \Theta). \tag{33}$$

The component density means are estimated as:

$$\boldsymbol{m}_i = \frac{\sum_{t=1}^{n} P(w_i|\boldsymbol{x}_t, \Theta) x}{\sum_{t=1}^{n} P(w_i|\boldsymbol{x}_t, \Theta)}. \tag{34}$$

The component density covariance matrices are computed as follows:

$$C_i = \frac{\sum_{t=1}^{n} P(w_i|\boldsymbol{x}_t, \Theta)(x - \boldsymbol{m}_i)(x - \boldsymbol{m}_i)^T}{\sum_{t=1}^{n} P(w_i|\boldsymbol{x}_t, \Theta)}. \tag{35}$$

Where in all of the previous equations

$$P(w_i|\boldsymbol{x}_t, \Theta) = \frac{p_i b_i(\boldsymbol{x})}{\sum_{k=1}^{c} p_k b_k(\boldsymbol{x})}. \tag{36}$$

Using the above equations, the EM algorithm now consists of the following steps:

Initialize Start with initial model parameters, Θ^o

E-Step Estimate new model parameters $\hat{\Theta}$ using the estimation equations

M-Step Replace current model parameters with the new model parameters

Iterate Iterate between the **E** and the **M** steps until the likelihood function stops increasing

Looking closely at Eqn. (36), which can be rewritten as

$$P(w_i|\boldsymbol{x}_t,\Theta) = \frac{|C_i|^{-1/2}\exp\left[-\frac{1}{2}(\boldsymbol{x}_t-\boldsymbol{m}_i)^T C_i^{-1}(\boldsymbol{x}_t-\boldsymbol{m}_i)\right]p_i}{\sum_{k=1}^{c}|C_k|^{-1/2}\exp\left[-\frac{1}{2}(\boldsymbol{x}_t-\boldsymbol{m}_k)^T C_k^{-1}(\boldsymbol{x}_t-\boldsymbol{m}_k)\right]p_k}, \tag{37}$$

it can be noted that when the Mahalanobis distance between $\boldsymbol{x}_t$ and the cluster mean, $\boldsymbol{m}_i$, is small and between $\boldsymbol{x}_t$ and the other means is large, the a posteriori probability is high (approaching the upper limit which is one). Using the approximation that

$$P(w_i|\boldsymbol{x}_t,\Theta) \approx \begin{cases} 1 & i = \arg\left[\underbrace{\min}_{j} D_M(\boldsymbol{x}_t,\boldsymbol{m}_j)\right] \qquad j = 1,\ldots,c \\ 0 & \text{otherwise.} \end{cases} \tag{38}$$

then Equations (33)–(35) become:

$$p_i = \frac{1}{n}\sum_{x\in\mathcal{H}_i} 1 = \frac{n_i}{n}, \tag{39}$$

where n_i is the number of data points in the i^{th} cluster. The component density means become:

$$\boldsymbol{m}_i = \frac{1}{n_i}\sum_{x\in\mathcal{H}_i}\boldsymbol{x} \tag{40}$$

and the component density covariance matrices are:

$$C_i = \frac{1}{n_i}\sum_{x\in\mathcal{H}_i}(x-\boldsymbol{m}_i)(x-\boldsymbol{m}_i)^T \tag{41}$$

which are the same equations used to calculate the weights, the means, and the covariance matrices in WMD.

This shows that the WMD method is actually an approximate method of finding the actual distribution of the data set as a mixture of Gaussian distributions. Its theoretical

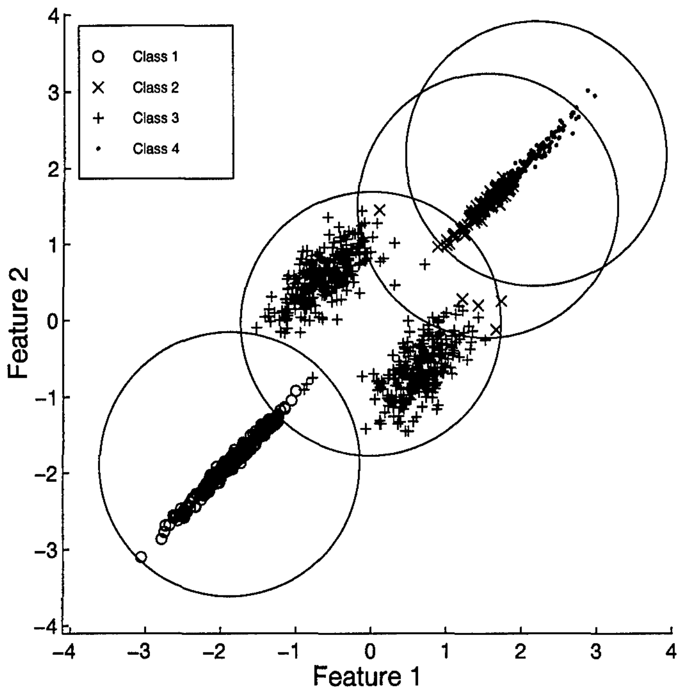


Figure 7. The four different Gaussians detected using the GMM/EM approach using initial parameters estimated from applying the LBG algorithm with Euclidean distance

justification is much stronger than using only the Euclidean distance as shown in Eqn. (36). It is a faster and much simpler technique to perform than the Expectation Maximization of Baum and yet attains accurate results as will be seen in Chapter *IV*.

One important issue in the GMM/EM approach is the need for an initial set of parameters. One way to come up with the initial model is to run a clustering algorithm on the data and estimate the mean, covariance matrix, and prior for each component of the mixture. In fact GMM/EM was not able to come up with the correct four Gaussian clusters of data in Fig. 7 because it used LBG clustering with Euclidean distance to find the initial partition from which the initial model, Θ^o, was estimated. The circles in the Fig. 7 are the equi-Euclidean distance from the estimated means of the Gaussians. It is clear that it made the two fat clusters into one elongated Gaussian cluster. WMD does not have that problem because it starts from the raw data.

It is worthwhile to mention that the EM update equations reappeared in the literature as an optimal fuzzy clustering [14], where the authors defined the quantity $\frac{1}{p_i b_i(\boldsymbol{x}_t)}$ as an "exponential" distance measure $d_e^2(\boldsymbol{x}_t, \boldsymbol{m}_i)$ and the component covariance matrices were called *fuzzy covariance matrices.* In this paper, Gath and Geva realized that the initial model selection has an important effect on the overall results. Therefore, they grew the codebook to the desired number of codewords, one codeword at a time, by a procedure that places the new codeword in a region where the data points have low degree of membership in the existing clusters. This is similar to the way splitting in the direction of maximum variance would ensure systematic distribution of the codewords.

The optimal fuzzy approach gave very good results that captured the shape of underlying substructures in many cases including the Iris data set, linear distributions, and clustering of sleep EEG recordings in order to classify the signal into various sleep stages. However, the required computation associated with calculating the a posteriori probabilities for each pattern after every iteration can become exorbitant.

2.7 Summary

This chapter has provided a background on clustering and has presented the theoretical justification for the approaches used in this thesis. The key ideas are summarized as follows:

1. Some clusters fall naturally in hyperellipsoids in the feature space rather than in hyperspheres.
2. The WMD algorithm allows for individually shaped hyperellipsoidal clusters with constraints on the covariance matrices.
3. The WMD algorithm is related to the expectation maximization algorithm.

III. Methods

3.1 Introduction

Having discussed in Chapter *II* the concepts used in the four clustering techniques (LBG, FCM, WMD, and HEC), this chapter presents the details of implementing these methods. First, Section 3.2 discusses the general approach followed in batch k-means-type clustering with variations in all stages of the algorithm. Next, Section 3.3 describes the implementation of the LBG algorithm. Next, Section 3.4 describes how the FCM approach was implemented. In Section 3.5, the details of the weighted Mahalanobis distance algorithm are presented. The first three methods are described together because they are similar in following the general structure of the k-means algorithm, so they may be implemented together to realize significant savings in computation and time. Section 3.6 outlines the methodology used in building the neural network for performing the principal component analysis. Finally, Section 3.7 describes the HEC network for integrating the Mahalanobis distance in partitional clustering.

3.2 A General Approach for Batch k-means Clustering

As discussed in Chapter *II*, the LBG, FCM, and WMD techniques are all based on the generalized Lloyd algorithm. This section presents the general approach taken by these algorithms, which is shown in Fig. 3.2.

After we acquire the data and preprocess it, an initial partition is needed before the iterative process starts. Each clustering epoch consists primarily of the following

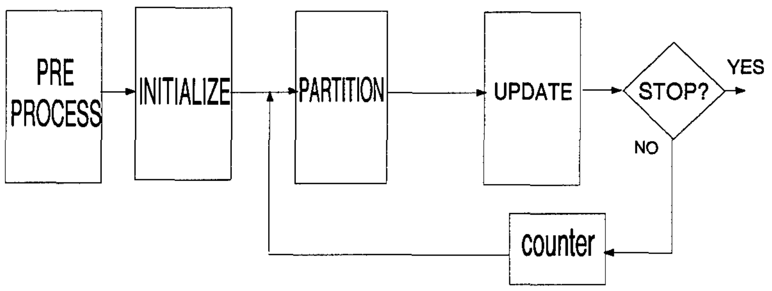


Figure 8. Flowchart of the general approach to perform batch k-means clustering

steps: partitioning according to a specific distortion measure, codebook update based on the results of the first step, and finally, evaluation of the current partition quality before another iteration is initiated.

The algorithms described here are implemented in MATLAB®. Appendix A contains the code named CLUSTER used to implement the LBG, FCM, and WMD techniques. A description of its operation and a list of its subroutines are also provided. The following sections will describe each stage of this general code.

3.2.1 Preprocessing. A normalization procedure that is composed of two steps is performed for all cases. First, the mean of the data set is subtracted from it which is equivalent to forcing the mean of the input patterns to equal zero. This is especially important for the PCA network. Second, we scale all the data sets to make the maximum length of feature vectors to be one without changing the shape of clusters or the relative location or ratio between the features. By doing so, the same set of parameters can be applied to all the data sets, and any initial codebook with small length is guaranteed to be in the middle of the data points in the feature space.

3.2.2 Initialization. As stated in Section 2.6.4, the choice of the initial model is critical for the quality of the results. The parameters needed to start each epoch of the iterative process depend on the clustering algorithm under study. Nevertheless, all clustering techniques require an initial codebook to improve. Therefore, this section will describe the different ways of finding the initial codebook while initialization for different parameters required for the Mahalanobis distance-based algorithms is discussed later.

A variety of codebook initialization schemes have been investigated in hopes of providing accelerated convergence of the k-means algorithm, achieving a better local minimum, and providing flexibility in terms of the number of clusters. In the following sections, some of the techniques that are incorporated in CLUSTER are described.

3.2.2.1 Initialization by Splitting. Linde *et al.* [24] suggested a splitting method whereby one starts by finding the centroid of all the training data which is the optimum codebook of size one. This single codeword, m_o, is then split to form two codewords as follows:

$$\boldsymbol{m}_{1,2} = \boldsymbol{m}_o \pm \boldsymbol{\rho}, \tag{42}$$

where $\boldsymbol{\rho}$ is a small perturbation vector. The clustering algorithm is then run to get a good codebook of size two. The design continues in this way in stages; the final code of one stage is split to form an initial code for the next.

Sometimes, the splitting technique is desired to grow the codebook, but the number of clusters, k, is not an integer power of two. For such cases, a modification of the method is implemented wherein the codebook is grown by splitting up to the largest N codewords such that,

$$N = 2^i \leq k, \qquad i = 1, 2, \ldots.$$

At that point, the codewords are sorted in descending order according the population of each cluster and only the first $(k - N)$ clusters are split.

3.2.2.2 Fixed Codebook Size Initialization. Starting with a codebook of the right size has the advantage that the computations associated with growing a full-sized codebook by splitting are eliminated. Thus, the design takes less time to complete and also allows an arbitrary number of codewords in the final codebook.

Perhaps the simplest fixed codebook size initialization technique that is used in the k-means algorithm [33] is to use the first k vectors in the training sequence as the initial prototypes. Another simple approach is to generate small random vectors as the initial codebook. These initialization methods are inefficient since some of the initial codewords may lie in a region with no patterns, and thus result in some unused codewords.

Bezdek *et al.* [5] proposed another initialization method that disperses initial prototypes uniformly along each feature range starting from the point $\boldsymbol{v}_1 = (m_1, m_2, \ldots, m_d)^T$ to the point $\boldsymbol{m}_k = (M_1, M_2, \ldots, M_d)^T$, where m_i and M_i are the minimum and the maximum values of the i^{th} feature for all the data points, respectively. This method is referred to as *gridding* in CLUSTER.

Katsavounidis *et al.* [21] suggested that widely separated data points are likely to belong to different classes. Hence, they proposed a *Maximin* initialization algorithm based on the following steps: First, calculate the norms of all vectors in the data set. Choose the vector with the maximum norm as the first codeword. Then, with a codebook of size i, we compute the distance between any remaining training vector $\boldsymbol{x}_j$ and all existing codewords and call the smallest value the distance between $\boldsymbol{x}_j$ and the codebook. Then, the training vector with the largest distance from the codebook is chosen to be the $(i+1)^{th}$ codeword. The procedure stops when we obtain a codebook of size k.

DeSimio [11] used the Karhunen-Loève (KL) transform to position initial codewords along the directions of maximum variance. The number of codewords in these directions are chosen proportional to the variance in that direction. Thus, for directions where the data have relatively large variances, more codewords are used than for directions where the variances are small.

The initial codebook generation method begins by specifying the number of codewords to be developed. The covariance matrix, W, of the d-dimensional training data is then computed along with the associated eigenvalues, ψ_i, and eigenvectors, $\boldsymbol{e}_i$, for $i = 1, \ldots, d$.

The number of codewords along each eigenvector direction, k_i, is found according to

$$k_i = \left\lfloor \frac{\psi_i}{\sum_{j=1}^{k} \psi_j} + 0.5 \right\rfloor k. \tag{43}$$

where k is the desired codebook size. Codewords are formed by scaling the eigenvectors such that the k_i codewords are uniformly distributed over $\pm 2\sigma_i$. Where σ_i is the standard deviation in the direction of the i^{th} eigenvector.

To demonstrate the effect of initialization on the overall performance of a clustering algorithm, the LBG algorithm with Euclidean distortion measure was used to partition the Daisy data set, shown in Fig. 9, into eight clusters. Table 1 shows the initial sum of squared distortion using the initial codebook generated from the different techniques, the final distortion after the LBG algorithm converged, and the number of iterations required for convergence.

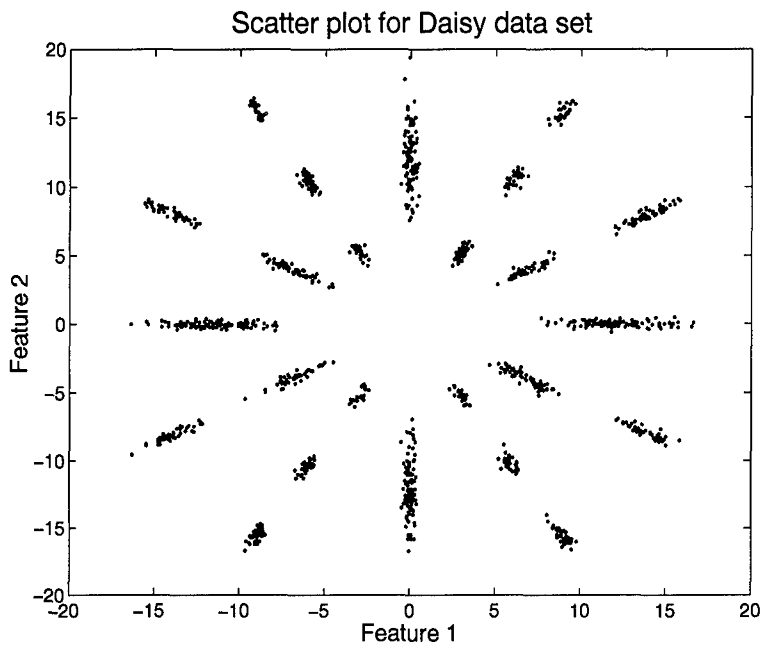


Figure 9. Scatter plot for the Daisy data set

Table 1. Table of the required initial and final distortion and the number of iterations required by the LBG algorithm to cluster the Daisy data set into eight clusters

Initialization Method	Initial Distortion	Final Distortion	Number of Iterations
Splitting	21550	15880	17
First-k-samples	295700	26430	7
Maximin	79710	15960	12
Gridding	97890	14690	8
KLI	41860	13610	7

As can be seen from the results, the worst is the "First-k-samples" approach because all the initial codewords may be of the same cluster. Splitting took more iterations to give better initial distortion, but it is not guaranteed to give the best final distortion.

For the purpose of simulations in the next chapter, LBG's splitting was used. For the cases where the required codebook size is not an integer power of two, the KL initialization (KLI) technique was preferred to the other techniques as it gives the best overall results as shown in Table 1.

3.2.3 Partitioning. The partitioning step consists basically of assigning each data point to the cluster whose centroid is nearest to that data point in d space, where d is the dimensionality of the data set. The minimum distance criterion can be used to determine the point membership of each centroid.

Hence, partitioning of X into k clusters is better described by a n x k matrix, U, whose entry, u_{ij}, is the membership of the i^{th} pattern in the j^{th} cluster. Therefore, the entries of U are numbers between zero and one and the sum of the entries in each row is one.

Note that in hard clustering the range of u_{ij} is $\{0,1\}$, which means that in cases of equal distances to more than one cluster center a tie-breaking rule is needed. In CLUSTER, the least numbered cluster is considered a winner.

3.2.4 Codebook Update. There are two ways for updating cluster centers, the sequential approach, where the codewords are updated after each sample, and the batch method, where the update takes place after all the data points are assigned to the different cluster centers. Even though the HEC algorithm adjusts the weights adaptively to estimate the eigenvectors and eigenvalues, the actual update to the cluster centers are done by a batch vector quantization algorithm [25]. Therefore, we restrict our analysis in this thesis to the batch mode.

3.2.5 Clustering Evaluation. The final logical step within an iteration is the evaluation of the clusters produced by the previous steps. Once formed, clusters must be evaluated to determine if the present configuration is optimal or whether modifications are necessary.

The following sections discuss in detail the implementation of each of the three clustering algorithms. For each technique, the specifics about which initialization technique is used, what distance measure is implemented, how to perform the update, and what criterion for cluster evaluation is necessary are clearly identified.

3.3 The LBG Algorithm Implementation

We followed the algorithm described in [24] using the Euclidean distance as a basis for partitioning. The codebook was grown by splitting, where a fixed small random perturbation vector is added to each codeword after each splitting stage.

The n x c distance matrix containing the Euclidean distance from the data points to each codeword is calculated using Eqn. (6). The optimality criterion is the mean squared error (MSE). The algorithm is terminated when (MSE) stops decreasing significantly.

3.4 The Fuzzy c-means Algorithm Implementation

There is no specific initialization technique suggested for the FCM algorithm [7]. Bezdek suggested starting with a random codebook but for the purpose of comparison, the same method for initialization was used for all algorithms. The Euclidean distance was also used as a basis for partitioning to compare with the Mahalanobis distance-based algorithms.

While in LBG only data points which are assigned to a specific cluster will update its codeword, in FCM all the data points will affect the update of the codewords. In other words, every data point is assigned to all the codewords but with different degrees. Recall that if $u \in \{0,1\}$ the calculation of M reduces to computing the means for the different clusters. Hence this same equation, Eqn. (8), is used for updating the cluster center for both the LBG and the FCM algorithms.

Different values for the exponential weight, m, were tried. It was found that as m gets larger, the effect of far points on the cluster center becomes clearer by shifting its location towards them. Keller *et al.* found that, in general, a value of $m = 2$ is appropriate [22]. However, a value of $m = 3$ was used for the simulations to allow for more fuzziness.

When using the fuzzy membership functions to perform classification, one considers the degree of class membership for each test vector. The class membership function values, U, computed with Eqn. (9), provide a measure of similarity by which a decision can be made to assign the pattern to the class that yields the highest membership value.

3.5 The WMD Algorithm Implemetation

The algorithm starts with measuring the mean and covariance of the whole data set. As suggested by [24] the required c codewords can grow by splitting. Splitting can be done randomly or according to other considerations. However, we think that the best splitting is that which will put the new codewords for a specific cluster far apart in the direction of the maximum variance of that cluster. We made the perturbation for each cluster based on eigenvalues and eigenvectors of its covariance matrix as

$$\boldsymbol{\rho}_i = \frac{1}{\sqrt{\psi^i_m}} \boldsymbol{e}^i_m \tag{44}$$

where ψ^i_m and $\boldsymbol{e}^i_m$ are the maximum eigenvalue and the corresponding eigenvector of C_i, respectively. The benefit of this approach is that there is a systematic approach of distributing the codebook vectors and each cluster will split in different directions based on the distribution in that cluster independent from other clusters.

In cases where the number of codewords is not an integer power of two, the KLI algorithm was used with the covariance matrix for all data points is used as C_i for all clusters.

Two ways to compute the Mahalanobis distance are possible, using C_i^{-1} as in Eqn. (4) or using the eigenvectors and eigenvalues of C_i as suggested by Eqn. (17). The second method avoids the need to compute the distance to each data point separately so matrix multiplication can be used instead of the loops which results in huge savings. Moreover, the second method turns away from inverting the different covariance matrices after each iterations and with proper implementation, we can keep the eigenvalues and eigenvectors so we can also obtain the new codewords after each splitting stage as the byproduct.

After all the data is presented, the codeword for each cluster will be computed as the average of the data points assigned to that cluster. The covariance matrix of these data points forms the A matrix for that group. The weight, w_i, can be calculated as the ratio of the number of samples in the i^{th} cluster to the size of the training samples for the WMD with conscience (WMDC). For the WMD with volume constraint (WMDVC)

$$w_i = \frac{1}{|C_i|^{\frac{1}{d}}} \tag{45}$$

where $|C_i|$ is the determinant of C_i.

The algorithm is terminated when the difference between the location of the codewords in consecutive iterations falls below a threshold or J_{GMSE} stops decreasing significantly for the case of WMDVC.

3.6 Principal Component Analysis Network

The proposed neural network was built as shown in Fig. 5. In using the update equations, Eqn. (20) and Eqn. (21), it should be noted that it is required to have the mean of the input patterns equal zero. In addition, the weights should be normalized after every update.

The weight vector $\boldsymbol{w}_1$ converges to the eigenvector with the largest eigenvalue of the covariance matrix C of the pattern set. Subsequently, the eigenvector corresponding to the next largest eigenvalue is computed and so on up the number of the output nodes.

This network was able to find the eigenvectors for the covariance matrix of several data sets accurately and quickly. However, it was noticed that this algorithm doesn't converge easily when the eigenvalues are small. A modification of the learning rules was implemented by Mao and Jain [25]. However, since the authors didn't have theoretical justification and introduced *ad hoc* parameters that might not generalize to other data sets, the modified learning rules were not implemented.

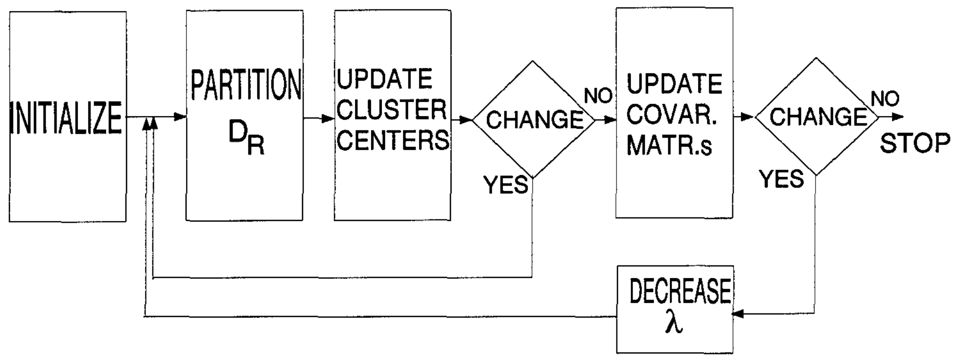


Figure 10. Flowchart of the HEC algorithm

3.7 Hyperellipsoidal Clustering Algorithm

Since the PCA network didn't converge easily especially when the eigenvalues were small and since there are exact methods for computing the eigenvectors and eigenvalues of a matrix, the same algorithm was implemented but using MATLAB's **eig** function to find the eigenvectors and eigenvalues after each iteration. The algorithm is depicted in Fig. 10.

For initialization, HEC starts with the identity covariance matrices for all clusters and random cluster centers. The value of λ starts with unity and it decreases by a specific amount, $\Delta\lambda$ until it reaches a specified minimum value, λ_o. $\Delta\lambda$ and λ_o are chosen to be 0.2 and zero, respectively, similar to what was used in [25]. The value of ϵ on the other hand is chosen to be 10^{-6}.

An important characteristic of the HEC algorithm is the inner loop where the cluster centers may change location without updating the covariance matrix (shape of the equi-Mahalanobis distance ellipse). This was shown to result in saving in computations as it avoids the need to recalculate the covariance matrix and it also provided some improvement in the results.

The stopping criterion used in both loops is whether or not there is a change in the partitioning as long as the total number of iteration is below a threshold.

3.8 Summary

This chapter has presented the methodology used in each of the LBG, FCM, WMD, and HEC algorithms for clustering. The general structure for a batch k-means was introduced with a description of each stage of this general algorithm. A neural network for principal component analysis was discussed. The following chapter presents results obtained by applying these techniques to several pattern recognition and Image coding problems.

IV. Results

4.1 Introduction

In this chapter, we investigate the performance of the WMD algorithm on several data sets. Comparisons are made with the LBG algorithm of Linde *et al.* [24] and the fuzzy c-means (FCM) as outlined in Bezdek [7], both of which use Euclidean distance. The algorithm for hyperellipsoidal clustering (HEC) [25] which uses a regularized Mahalanobis distance is also compared.

Each of the four cluster analysis techniques described in the previous chapter is applied to several test cases. The first set of tests utilizes two-dimensional Gaussian distributions, which are easy to both visualize and analyze. The results of these tests is described in Section 4.2. Section 4.3 shows the results of applying the different algorithms on four-dimensional measured samples of the Iris flower. These sets of cases are the same as those used by Mao and Jain in their paper [25]. Some of these test cases are generated by taking samples from known classes so that the true error rate may be computed to check the accuracy of the classification provided by each technique. The performance of these algorithms on image coding vector quantization is also studied in Section 4.4 as a practical application.

In all cases, the performance metric is not minimum mean squared error (MSE). It is well known that MSE does not imply highest classification accuracy or highest perceived visual quality [12]. Indeed, in all cases, WMD codebooks resulted in larger MSE than the Euclidean-based codebooks. However, WMD clustering, classification and perceptual quality performance are shown to be superior to the competing methods.

For the cases where the codebook size is not an integer power of two, clusters of the training samples are formed using the Karhunen-Loève initialization [11]. The Karhunen-Loève initialization was chosen to initialize the codebooks because it disperses the desired number of codewords along the principal component axes of the data's covariance matrix, allowing simple and efficient implementation of *any* number of codewords.

Table 2. Results of clustering the 2-dimensional mixture of Gaussians data into three clusters by the different clustering methods

Clustering Algorithm	# of Errors	# of Iterations	Megaflops
LBG	231	5	2.89
FCM	204	20	4.67
WMD (with conscience)	3	8	3.29
WMD (volume constraint)	5	14	3.81
HEC	3	20	4.37

4.2 Two-Dimensional Test Cases

The WMD algorithm was used to design a quantizer for 2-dimensional data with four Gaussian distributed clusters, see Fig. 11(a).

LBG clustered the data into four clusters as shown in Fig. 11(b). On the other hand, FCM did converge to the desired four clusters after 15 iterations as shown in Fig. 11(c). The WMD algorithm grouped the data points as shown in Fig. 11(d) after four iterations only. The large circles in Fig. 11(b) and 11(c) show the equi-Euclidean distance points from the means. The equi-Mahalanobis distance ellipses are shown in Fig. 11(d) to follow the shape of the actual clusters. This demonstrates the natural grouping of WMD.

The WMD algorithm was also used to design a quantizer for 2-dimensional data with three Gaussian distributed clusters, see Fig. 12(a). LBG divided the data into three classes after 5 iterations as shown in Fig. 12(c). FCM classified the data points as shown in Fig. 12(d) in 20 iterations. The WMD with conscience (WMDC) algorithm converged in 8 iterations to the clusters shown in Fig. 12(e) with only 3 misclassified samples as compared to the actual distributions that generated the data. This is similar to the results of HEC algorithm but with fewer iterations and less computations. WMD with volume constraint (WMDVC), on the other hand, resulted in 5 errors after 14 iterations. The large circles in Fig. 12(a) and Fig. 12(b) show the equi-Euclidean distance points from the cluster centers. The equi-WMD ellipses are shown in Fig. 12(c); note that they follow the shape of the actual clusters. We can see that WMD outperforms both LBG and FCM and would similarly outperform any clustering technique that uses Euclidean distance.

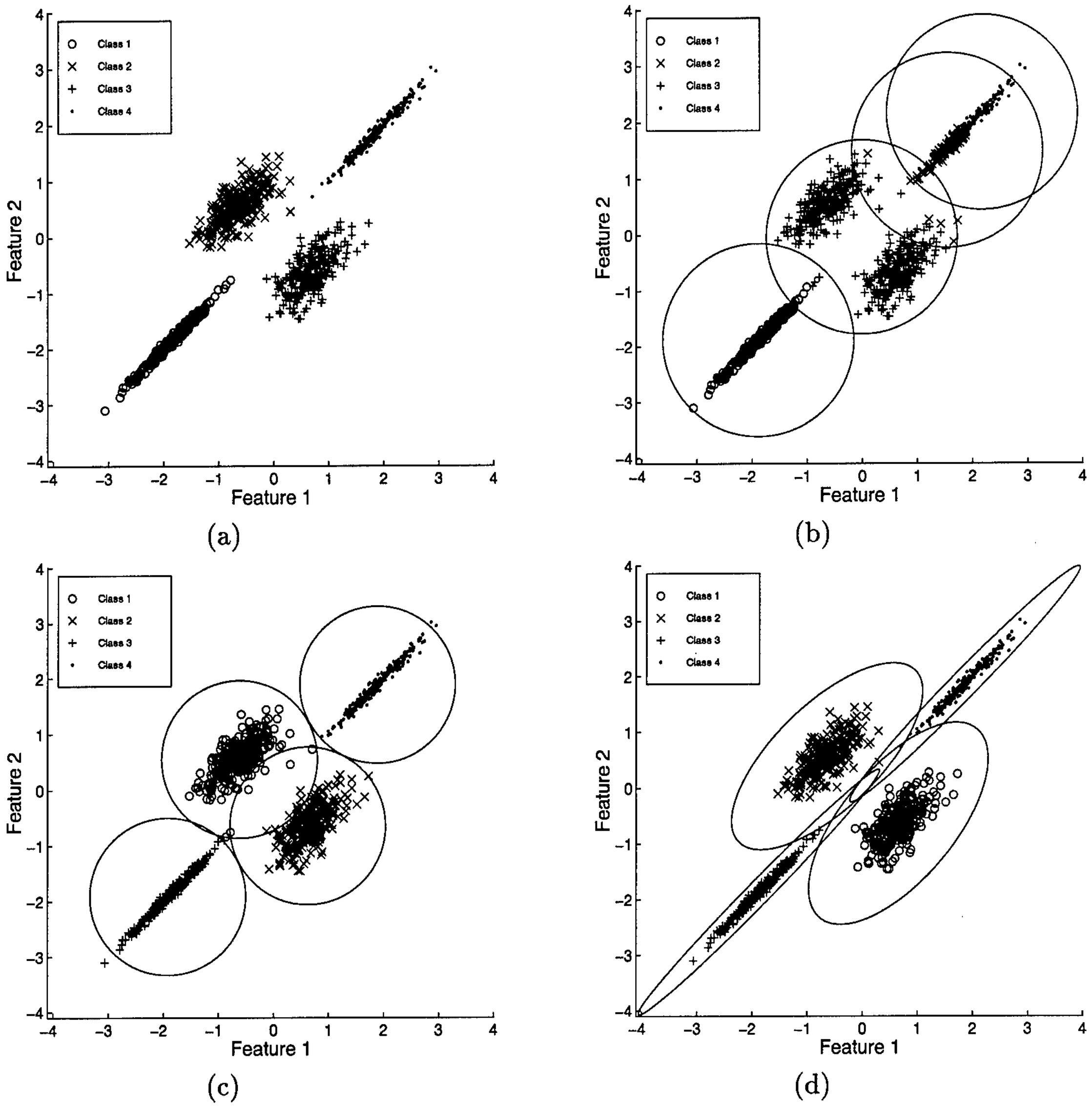


Figure 11. Gaussians data set (a) and location of cluster centers upon convergence (b) LBG, (c) FCM, (d) WMD.

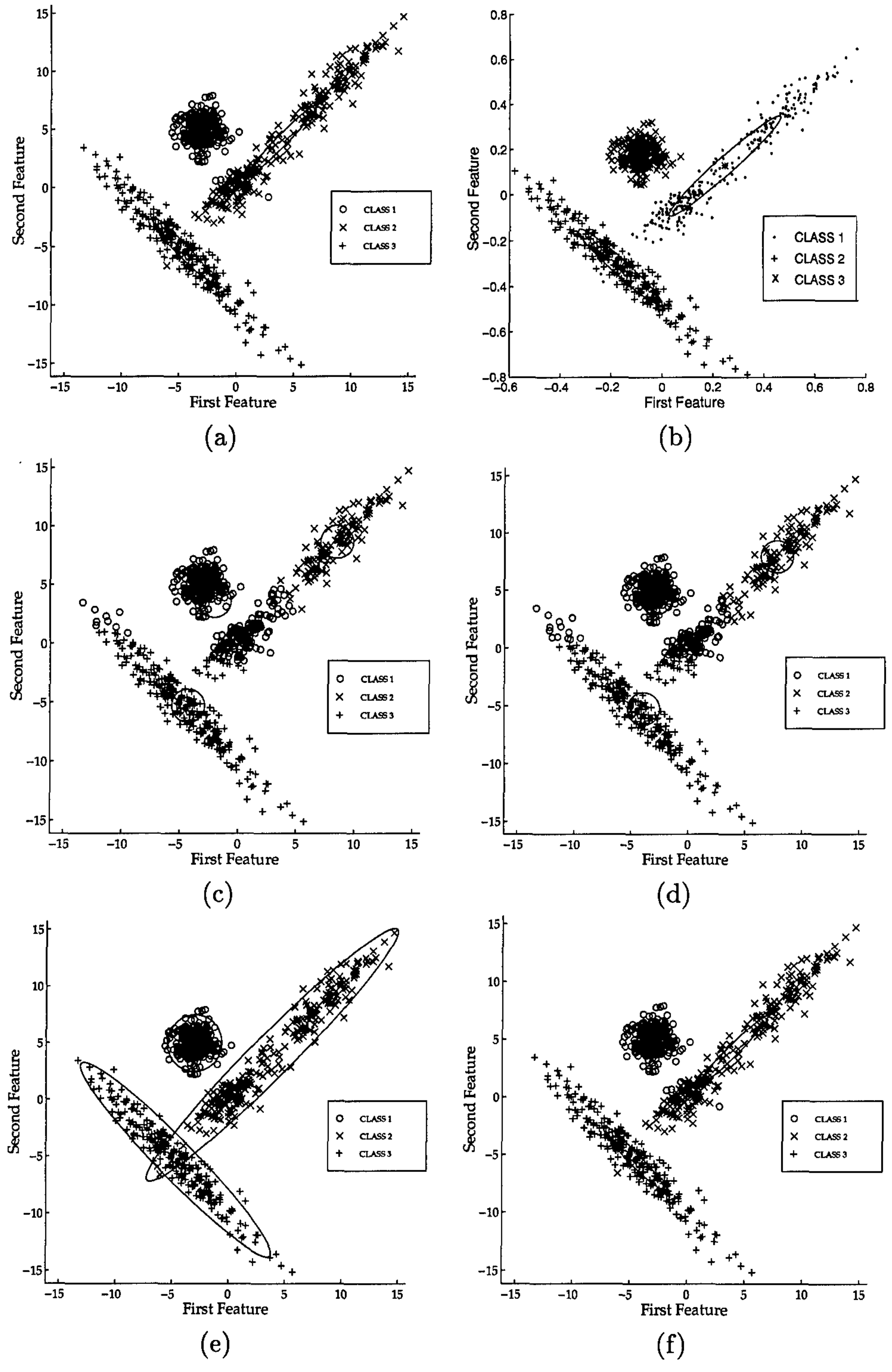


Figure 12. Original and computed partitioning of a mixture of Gaussian data using the different clustering methods. (a) Original classes. (b)–(f): Resultant One-nearest neighbor classification using the codebook generated by HEC, LBG, FCM, WMDC, and WMDVC methods, respectively.

Table 3. Results of clustering the Iris data into three clusters by the different clustering methods

Clustering Algorithm	# of Errors	# of Iterations	Megaflops
LBG	16	6	0.58
FCM	15	12	1.53
WMD (with conscience)	5	7	0.81
WMD (volume constraint)	7	13	1.41
HEC	5	21	2.03

Furthermore, Mahalanobis distance degenerates to Euclidean distance for hyperspherical clusters as shown by the circular cluster in Fig. 12(e) and Fig. 12(f).

4.3 Clustering of The Iris Data

We also applied WMD to the well-known Iris data set. This data set has four features representing three different species of the Iris flower. Figure 13(a) shows the projection of the Iris data on the two main principal components. Figures 13(b)-(f) show the results of minimum distance classification of the samples using the different techniques. Note that the projection of the Iris data on the first two principal components should give us a good idea about the shape of the clusters since the third and fourth eigenvalues are very small so the variation along those eigenvectors are small.

After 6 iterations, LBG resulted in misclassification of 16 data points. FCM resulted in 15 patterns to be misclassified after 12 iterations. On the other hand, WMDC resulted in only five errors after 7 iterations. Using the WMDVC resulted in 7 errors in 13 iterations. The WMDC result is consistent with the results of HEC [25] but achieved with our much simpler design.

The reason for this improved performance of the Mahalanobis distance-based clustering is that Iris classes are not well separated, and they do not occupy a hyperspherical shape in the feature space. These facts are easily demonstrated by Fig. 13(a). Comparing WMD with HEC, WMD eliminates the overhead of learning the network weights and does

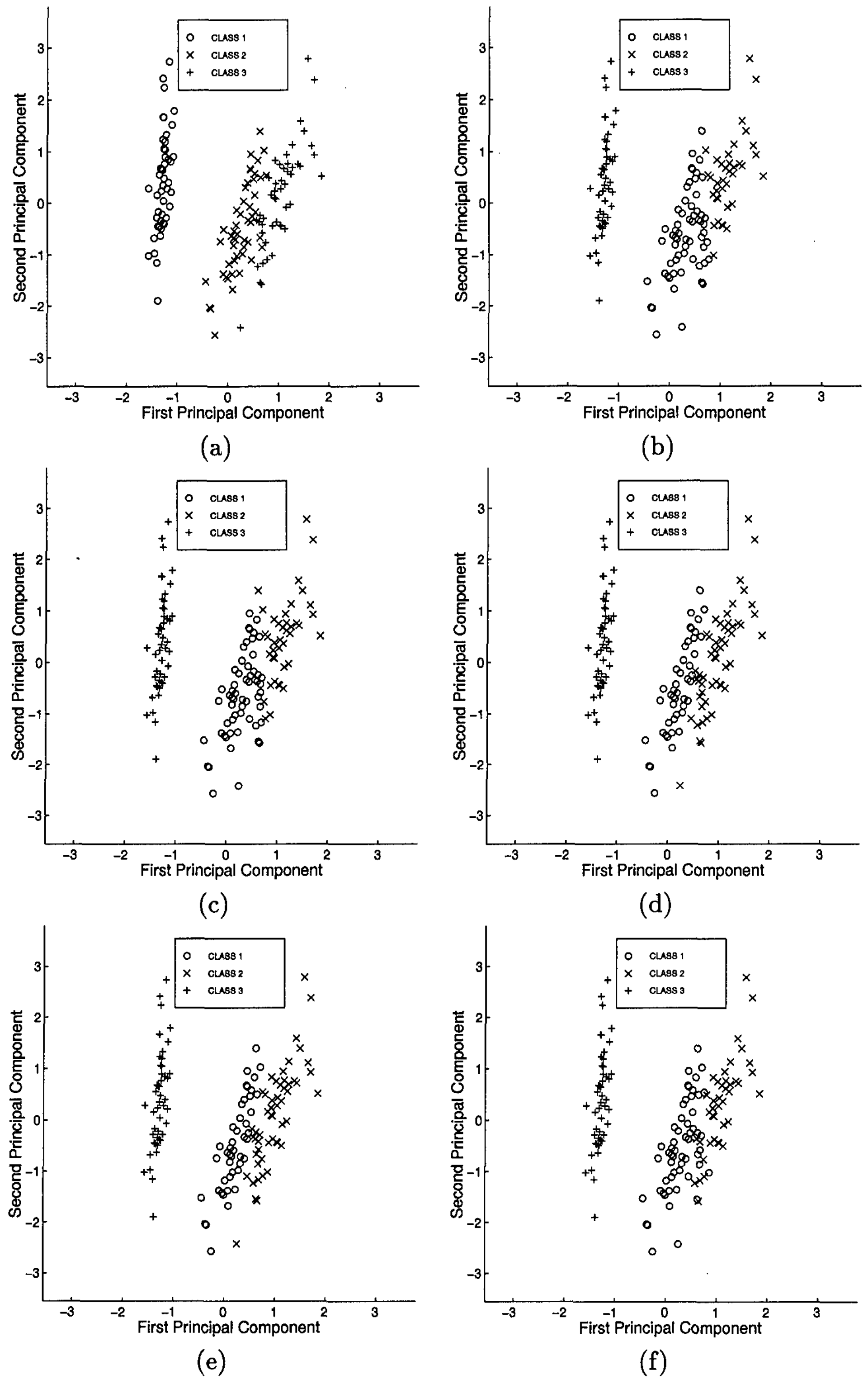


Figure 13. Original and computed partitioning of Iris data using the different clustering methods. (a) Original classes. (b)–(f): Resultant One-nearest neighbor classification using the codebook generated by LBG, FCM, HEC , WMD with conscience, and WMD with volume constraint methods, respectively.

not require an upper limit on the number of iterations (100,000 was used in HEC) [25]. Furthermore, no *ad hoc* parameters are required for WMD.

4.4 Vector Quantization Coding of Image Data

In this section, we evaluate the WMD clustering algorithm in a realistic application, specifically, vector quantization coding of an image. We have chosen this application, in part, because the computation is very intensive due to the large amount of data to be processed. In addition, the evaluation results can be expressed both numerically and visually, providing better insight of the differences among the algorithms.

Vector quantization image coding is used to reduce transmission bit rate or data storage while maintaining acceptable image quality. In this application, the mandrill image from the University of Southern California (USC) image database was used, see Fig. 14(a). It consists of 480 x 500 pixel digitized into 8-bit gray levels. Due to the large size of the image, it was down-sampled by a factor of four along both axes resulting in a more manageable problem. In this experiment we try to partition the images into 4 clusters and then define the centroids of resulting clusters as the codewords. Blocks of 4 x 4 pixels defining 16-dimensional vectors were then used to design the codebooks.

For the purpose of comparison, we test the performance of the WMD algorithm versus both LBG clustering and Fuzzy c-means on computing the codebooks for the images. Figures 14(b), 14(c), and 14(d) show the reconstructed images using the four-class codebook generated by WMD, FCM, and LBG clustering methods, respectively. Comparing these figures, we find obvious degradation of the visual quality in LBG and FCM reconstructed images especially in the nose area. Only the WMD image shows the three-level change in the nose pixels and the continuity of the dark gray level all around the nose. The covariance matrix of the pixels corresponding to this gray level was found to be different from the identity which implies a non-hyperspherical shape. WMD groups the vectors naturally, which provides the clear improvement in the quality of the reconstructed image. The proposed algorithm adapts to the local characteristics and preserves the features of the face.

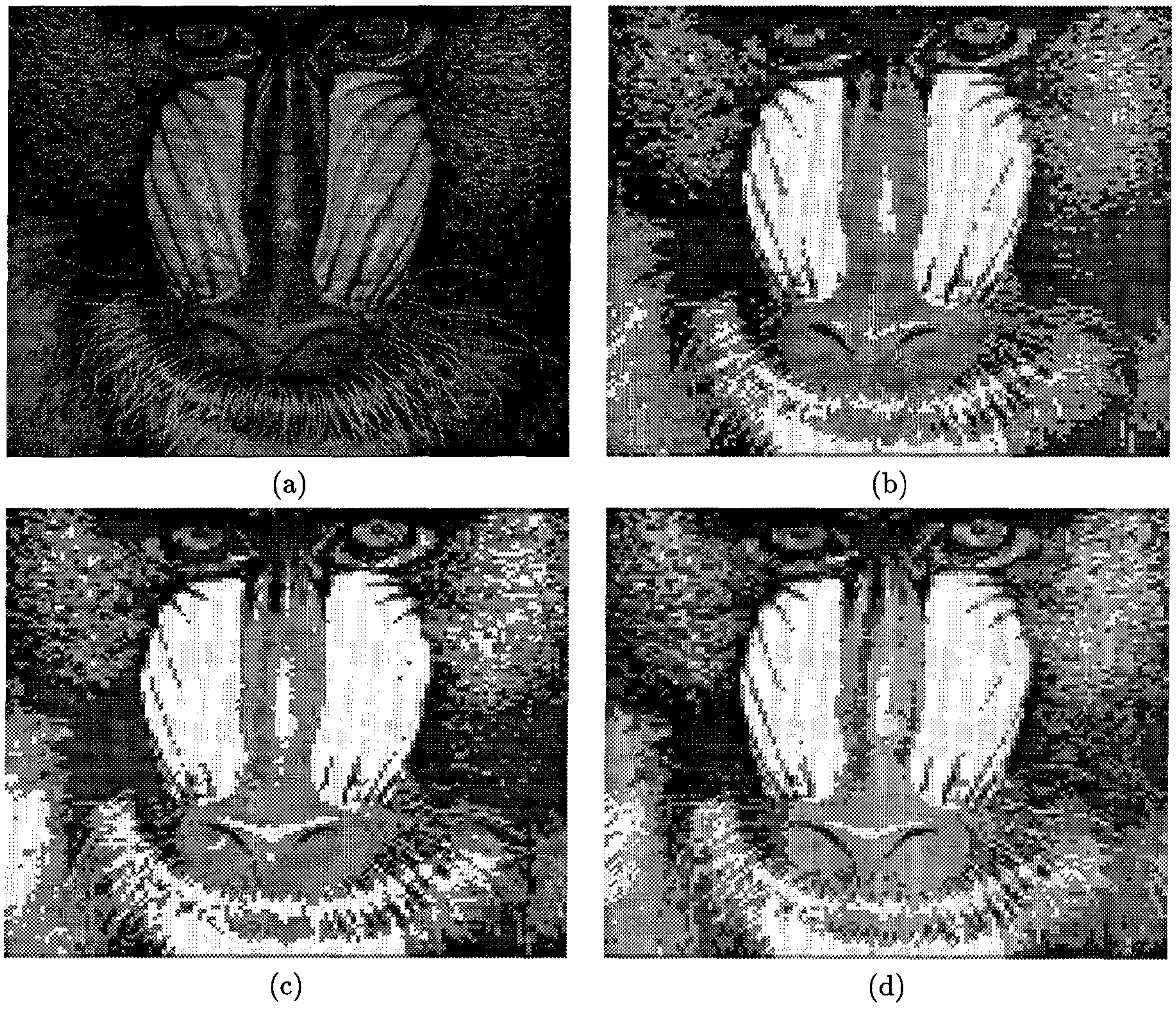


Figure 14. Original and reconstructed Mandrill images using the different clustering methods. (a) Original Mandrill image. (b), (c), and (d): Reconstructed image using the four-class codebook generated by WMD, FCM, LBG methods, respectively

The Lena image from the USC image database was also used, see Fig. 15(a). It consists of 480 x 512 pixel digitized into 8-bit gray levels. We used blocks of 4 x 4 pixels defining 16-dimensional vectors to design a four-class codebook.

Figures 15(b), 15(c), and 15(d) show the reconstructed images using WMD, FCM, and LBG clustering methods respectively. Comparing these figures, we find obvious degradation of the visual quality around the shoulders and faces in LBG and FCM cases. Only the WMD image shows the three-level change in the gray level in the face and the right-hand side of the background. WMD groups the vectors naturally which provides the clear improvement in the quality of the reconstructed image.

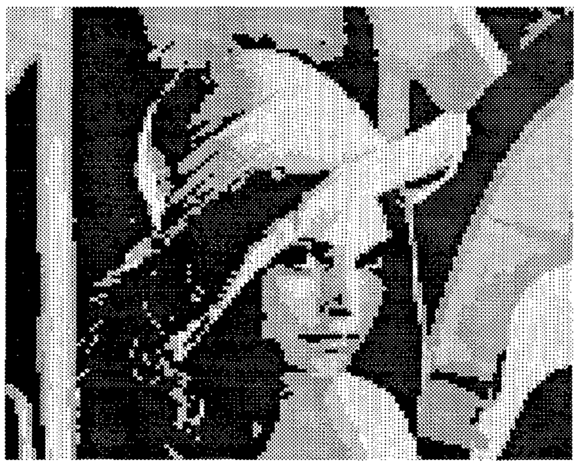
As can be seen in Fig. 15(c), it is possible in the FCM algorithm to have very close codewords that partition the space based on almost equal membership function values. Hence we see only one gray level and two white levels.

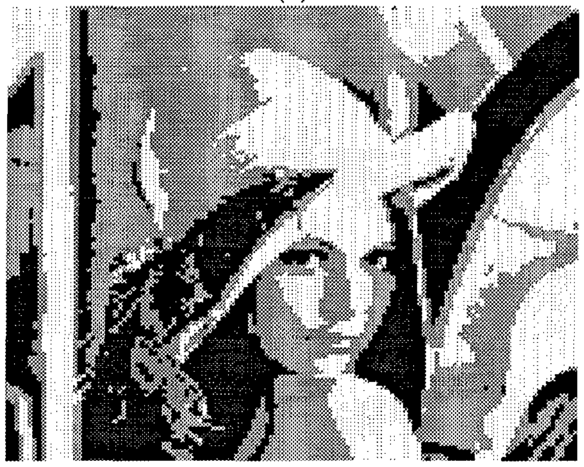
Trying to apply the HEC algorithm on some test cases, it was noted that one or no samples are assigned to the centroid of a specific cluster. Therefore, it is difficult to compute a covariance matrix for that cluster and the algorithm ends up with less than k clusters.

This problem can be attributed to two factors: First, random initialization may result in cluster centers that is farther away from the samples than the other clusters. Such case was encountered when HEC was applied to Daisy data set, see Fig. 9. Even though the authors didn't suggest a solution to this problem, this problem was fixed by splitting the cluster that has the largest number of data points. This solution solved the problem for the Daisy data set but when HEC was applied on the image vectors it failed to converge.

The failure to converge on the image coding problem and other cases points to the second possible cause of this behavior which is the problem of large clusters engulfing smaller ones. See problem three in Section 2.6.1. It is clear that after several iterations, the value of λ becomes zero and the regularized Mahalanobis distance reduces to Purely Mahalanobis distance without any constraint on the covariance matrix. The fact that this problem doesn't occur for the WMD method reassures the need for a constraint, such as the conscience or the volume constraint, when using the Mahalanobis distance in clustering.

(a)


(b)


(c)

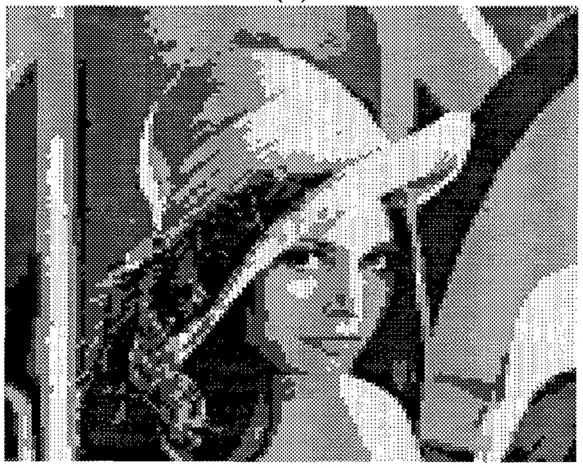

(d)

Figure 15. Original and reconstructed Lena images using the different clustering methods. (a) Original Lena image. (b), (c), and (d): Reconstructed image using the four-class codebook generated by WMD, FCM, LBG methods, respectively

4.5 Summary

This chapter has shown the results obtained by applying each of the four techniques to two two-dimensional and one four-dimensional test cases. In addition, vector quantization coding of two images were intried. In general, these results indicate that the Mahalanobis distance is a useful metric to be used as a similarity measure. Nevertheless, it was shown that there exists a necessity to control the covariance matrix of each cluster in order to obtain good results. The next chapter summarizes the results obtained in this investigation and provides some recommendations for further research.

V. Conclusion

5.1 Introduction

The primary objective of this research was to examine the use of the class-based Mahalanobis distance in clustering and develop a method to solve the problems associated with such a use. The second objective was to compare its performance to Euclidean distance based algorithms and other Mahalanobis distance based algorithms. Both of the objectives were met.

5.2 Contributions

- This thesis introduced a method for integrating the Mahalanobis distance into batch k-means-type algorithm. This algorithm has several attributes that make it well suited for more general clustering problems: First, the system is robust in that it achieved similar success in all of the test cases studied. Second, the algorithm is related to the expectation maximization process so it has the same theoretical foundation as the Gaussian mixture models. Third, it builds the codebook starting from the data itself using an improved splitting method that takes the different shapes of clusters into account.
- The analysis showed that the covariance matrices in the Mahalanobis distance based algorithms need to be constrained in order to reach a non-trivial solution. Two different constraints were discussed and implemented.
- As a tool for follow-on research, this thesis intentionally provides a significant amount of background information in the area of clustering. Chapter *II* was specifically written to assist a novice to the field in learning many of the clustering problem fundamentals. The software developed in this effort contains numerous selections of clustering techniques and parameters. This provides ease in application or replication of this work for further research.

5.3 Follow-on Research

The research discussed in this thesis is by no means exhaustive. As with any large undertaking, there are many areas left for further research. Some possible enhancements are listed below:

- Preliminary results showed that the mutual Mahalanobis distances between the cluster centers $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ using both C_i and C_j are good indicators of the similarity between the two clusters in terms of shape and orientation. These indicators can be the basis for merging similar clusters.
- Exploring the possibilities for finding the appropriate number of clusters, k, by means of merging clusters based on the Mahalanobis distance between their centroids and finding a monotonically decreasing criterion function as the number of clusters k increases, such as J_{GMSE} in WMD with volume constraint. The best number of clusters will correspond to the "knee" of the curve where further increase in k results in only a slight decrease in the criterion function.
- Developing a way to handle outliers which affect the covariance matrix and hence the overall performance of a Mahalanobis distance based algorithm.

5.4 Summary of Results

Mahalanobis Distance is a very useful tool which is believed to give a better measure of similarity than the Euclidean distance. It was shown that for certain data sets, the Euclidean distance based clustering algorithms do not group the data points naturally. The weighted Mahalanobis distance clustering, on the other hand, results in codebooks that represent the data better than codebooks generated with either the LBG or the FCM algorithms.

Even though the neural network for hyperellipsoidal clustering (HEC) uses the Mahalanobis distance, it failed to perform as well as the WMD algorithm. This is because it could not solve the problems associated with using the Mahalanobis distance in clustering.

The WMD algorithm has been demonstrated using both artificial data and real data to outperform existing methods in maintaining the natural distribution of the feature space.

It was also shown that WMD clustering accounts for variability in cluster shapes, cluster densities, and the number of data points in each of the subsets.

The techniques developed for this thesis should be applicable not only to other synthetic data, but to any data sets which are characterized by non-spherical cluster shapes.

5.5 *Conclusions*

The results obtained in this thesis lead to several interesting conclusions:

- Hyperellipsoidal clusters are more general and more likely to be encountered in real-life applications than hyperspherical clusters.
- Increased generality allows better classification and perceptual image quality than Euclidean distance based algorithms.
- Unlike HEC, WMD was able to overcome the difficulties associated with using the Mahanobis disance in clustering.
- WMD has been related to the EM algorithm for finding the parameters in a Gaussian Mixture Model

Appendix A. Clustering Codes

This appendix contains the MATLAB® codes which implement the LBG, FCM, WMD, and HEC approaches.

To allow for saving in time and computation, a system for clustering is generated with numerous selections of data sets, codebook size, distance metrics, initialization techniques, and partitioning rules (hard or fuzzy). Therefore, a user can study the results of using these algorithms with different parameters. In combining the algorithms together in CLUSTER, some redundant operations are eliminated, such as reloading the data, normalization, and codebook initialization. In fact, it is possible to have the code run all the algorithms with all different choices or parameters and save the results and figures with different file names without supervision. This automatic storing of output files and figures saves a lot of time and enables the user to either start looking at the results while the code is running or leave the code running and examine the results later.

A.1 cluster.m

```
function [clus_centr_new,A_inv,lisdist,asigned_class]=...
         Cluster(Data_choice,codbksize,hard_fuzzy,...
         Mpara,Distnce,init_clus,weight)

%[clus_centr_new,A_inv,lisdist]=...
Cluster(Data_choice,codbksize,hard_fuzzy,Mpara,Distnce,init_clus,weight)
%
% Capt Khaled Younis, RJAF
%
% Aug 26, 1996
%
% A clustering package that perform Hard/Fuzzy c-means
% using Euclidean/Weighted Euclidean/Mahalanobis distance
% Initialization by Splitting/KLI/Min-max/Gridding or Randomly
%%%%%%% Get parameters %%%%%%%%%%%%%%
```

```
% Data_choice= input('Data set ...
(1) XOR data (2) Gaussians (3) IRIS (4) Daisy (5) Mandrill ...
(6) uniform (7) flir (8) Lenna (9) sub-manifold (10) Linear: ');
% codbksize = input('Codebook size : ');
% hard_fuzzy = input('Clustering Algorithm ...
(1) LBG ( HCM) (2) Fuzzy C_means : ');
% init_clus = input('Codebook Generation  ...
(1) Splitting (2) Fixed size (first C samples) (3) Maximin ...
(4) Bezdek (5) Desimio (6) get-then-split (7) random : ');
% Distnce = input('Similarity Measure ...
(1) Euclidean Distance (2) Diagonal covariance...
(3) changing Mahalanobis : ');
% weight = input('Weighting ...
(0) No weight (1) Conscience (2) Determinant constraint : ');
% Mpara=0; % to set it zero if not fuzzy
%    if hard_fuzzy==2
% Mpara = input('M parameter for Fuzzy C_means : ');
% end
% end %if nargin
%-------- End get parameters
  if nargin==6
  weight=1;
  end
flops(0)
tic
rsq=2;
close all
plt=1;
%%%%%% Initalize parameters for the whole program %%
lisdist=[]; %To record the distortion vs. number of iterations
```

```
epsl=0.001; %required change rate in Distotion
epsl2=1e-6; %Required min change in cc location (fcn of data)
savplt=1; %(1) saves (0) Not to save the plots
savres=2;      %(0) No save ..
 (1) saves in ASCII ...
 (2) save in variables in .mat
M=1; %For splitting
cls_tot=zeros(codbksize,1);
%------ End Initialize parameters

%%%%%%% Get data %%%%%%%%%%%%%%%%%%%%%%
data_in=getdata(Data_choice);
[row,col]=size(data_in);
av=mean(data_in);
norm_data=data_in-ones(row,1)*av;
max_length=max(diag(norm_data*norm_data'));
data_in==norm_data./(ones(row,col)*sqrt(max_length));
asignd_class=ones(row,1);
data_cl=[asignd_class data_in];
covr=cov(data_in);
covr_all=covr;
mx_data=max(data_in);
mn_data=min(data_in);
mx=max(mx_data)+1;
mn=min(mn_data)-1;
%------------- end get data

%%%%%%% Suitable matrix for ellipse drawing %%%D:disnce
  if Distnce==1,
    elps_cov=reshape(eye(size(covr)),1,col^2);
  elseif Distnce==2,
    elps_cov=zeros(size(covr));
    elps_cov=reshape(diag(diag(covr),0),1,col^2);
  elseif Distnce==3,
    elps_cov=reshape(inv(covr),1,col^2);    % initially Global covariance matrix
    %elps_cov=reshape(eye(col),1,col^2);    % initially Euclidean distance
  else error('invalid choice')
  end % if dist
A_inv=elps_cov; %redundant for other than WMD
%------------------------------------------

% For automatic savings of the figures and the results
Which=[Data_choice codbksize hard_fuzzy Mpara Distnce init_clus weight]

  %%%%%% Get initial cluster centers  %%%%%%%%%%%
  if init_clus==1  %splitting
  A0=av;
  clus_centr_old=A0; %initial codebook vector (centroid)

  elseif init_clus==2  % C samples
  clus_centr_new=data_in(1:codbksize,:)+10^(-6);    % first c samples

  elseif init_clus==3  %Minimax
  %  eta=.2; % min percetage of the distance to last cc
  %  clus_centr_new=min_max(data_in,codbksize,eta);
   clus_centr_new=turbo_mm(data_in,codbksize)+eps;
  elseif init_clus==4  % Bezdek's Gridding
  dif_over_N=1/(codbksize-1)*(mx_data-mn_data);
for i=1:codbksize
clus_centr_new(i,:)=mn_data+(i-1)*dif_over_N;
```

```
end %for i
  clus_centr_new=clus_centr_new+10^(-6);


  elseif init_clus==5  %KLI
  clus_centr_new=kli(data_in,codbksize);


  elseif init_clus==6
  load ./autosaves/mandmah4splt2
  giv_clus_centr = clus_centr_new;
  load ./autosavespert
  giv_pert = pert;
  M=length(giv_pert(:,1));
  elseif init_clus==7  % random
  clus_centr_new=randn(codbksize,col);
  else error('invalid choice')
  end %if init_clus
  %-------------------------------------------------


while M<codbksize
  %### Start splitting loop ###################
  if init_clus==1 %splitting
    if M==1
    [vec,val]=eig(covr_all);
    pert=(vec*sqrt(diag(val)))';
    end
  pertlbg=(randn(1,col)-.5)*1e-1*mean(mean(data_in));
  %%%%% Get cc & pert after splitting %%%%%%%%%%%%%%%
    if 2*M<=codbksize % Splitting by two
if Distnce ==3
clus_centr_new=...
```

```
[clus_centr_old+pert/M/2;clus_centr_old-pert/M/2];
else
clus_centr_new=...
[clus_centr_old+ones(M,1)*pertlbg;...
clus_centr_old-ones(M,1)*pertlbg];
end
    A_inv=[A_inv;A_inv];
    M=2*M;
    elseif 2*M>codbksize % add required number
    num_sp=codbksize-M;
    [Srt,IND]=sort(cls_tot);
if Distnce==3
clus_centr_new(IND(M:-1:M-num_sp+1),:)=...
clus_centr_old(IND(M:-1:M-num_sp+1),:)+...
pert(IND(M:-1:M-num_sp+1),:)/M/2;
clus_centr_new=[clus_centr_new;...
clus_centr_old(IND(M:-1:M-num_sp+1),:)-...
pert(IND(M:-1:M-num_sp+1),:)/M/2];
else
clus_centr_new(IND(M:-1:M-num_sp+1),:)=...
clus_centr_old(IND(M:-1:M-num_sp+1),:)+...
ones(num_sp,1)*pertlbg;
clus_centr_new=[clus_centr_new;...
clus_centr_old(IND(M:-1:M-num_sp+1),:)...
-ones(num_sp,1)*pertlbg];
end
    A_inv=[A_inv;A_inv(IND(M:-1:M-num_sp+1),:)];
    M=M+num_sp;
    end % if 2*M
    %------ End Get cc & pert
```

```
  elseif init_clus==6
  pert=giv_pert;
  clus_centr_old=giv_clus_centr;
  else
          M=codbksize;
          A_inv=ones(codbksize,1)*reshape(A_inv,1,col^2);
  end %if init_clus
  %------------ End splitting loop

  %%%%% find suitable A mtx for distance calc %%%
  if Distnce==1 %Euclidean
  A_matrix=ones(col,M);
  elseif Distnce==2 %Diagonal cov (Ecld divided by std)
  A_matrix=diag(covr)*ones(1,M);
  end % if Dis
  %-------------------------------------------------

%%%%%%%%% initialization for distortion calc loop
m=0; %counter for iterations each sp-stag
oldmxmov=0;var=0; %to control the number of iterations
Dist_old=realmax; %initialize distortion each sp-stage
Dist_new=99^49;
mx_movcc=100; %To check stability of moving cc in WMD
Distance=zeros(row,M); % Distance matrix
list=[];
consc=ones(1,M);
  %######### Start Distortion Calculations loop ############
      while ((Dist_old-Dist_new)/Dist_new>epsl) & mx_movcc>epsl2
  clus_centr_old=clus_centr_new;
  U=zeros(row,M); % Membership matrix update each iter
  i=0;
    %%%% find new distance matricies then U mtx %%%%%%%
    while i<row
    i=i+1;
if Distnce ~=3
Distance(i,:)=sum(((clus_centr_new'-(data_in(i,:))'...
*ones(1,M)).^2)./A_matrix);
else
  for k=1:M
  Distance(i,k)=[clus_centr_new(k,:)-data_in(i,:)]*...
reshape(A_inv(k,:),col,col)*...
[clus_centr_new(k,:)-data_in(i,:)]';
  Distance_wocon(i,k)=Distance(i,k)/consc(k);
  end %for k
end %if Dis

if hard_fuzzy==1  % HCM ( LBG )
Mpara=3;
[nn,ii]=min(Distance(i,:));
U(i,ii)=1;
elseif hard_fuzzy==2 % Fuzzy C_means
  for j=1:M
  U(i,j)=1/sum((ones(M,1)*sqrt(Distance(i,j))./...
 sqrt(Distance(i,:)')).^(2/(Mpara-1)));
  end % for j
else error('invalid choice')
end % end if hard
    end  % end for i
    %---------- end Distance & U calc
```

```
   %%%% Update the covariance mtx, pert, clus-centr %%%%%%%%%%
  [dummy,asignd_class]=max(U');
          clear dummy
  clus_centr_new=((1./sum(U.^Mpara))'*ones(1,col)).*...
 ((U.^Mpara)'*data_in);
    for p=1:M
    indx=find(asignd_class==p);
    len_class=length(indx);
    cls_tot(p)=len_class;
if len_class>1
covr=cov(data_in(indx,:));
  if len_class<col, covr=covr+eps*eye(col); 'eps', end
inv_covr=inv(covr);
  if Distnce==3
    if weight==0
    consc(p)=1; % no conscience
    elseif weight==1
    consc(p)=len_class/row; % conscience
    elseif weight==2
    consc(p)=1/(det(inv_covr))^(1/col); % volume constraint
    end
  A_inv(p,:)=consc(p)*reshape(inv_covr,1,col^2);
  det(reshape(A_inv(p,:),col,col));
  end
else
covr=covr_all;
end %if len
    [vec,val]=eig(covr);
    [mval,inval]=max(diag(val));
```

```
              pert(p,:)=sqrt(mval)*vec(:,inval)'; % mx vrnce dirctn
     %      pert(p,:)=(vec*sqrt(diag(val)))';    % sum of scaled e-vec
     end % for p
     if hard_fuzzy==1
     ind=find(sum(U)==1 | sum(U)==0);
     len=length(ind);
if len>0
[mx1,ind1]=max(sum(U));
clus_centr_new(ind,:)=(randn(len,1)+1)*clus_centr_new(ind1,:);
end %if len
     end        % end if hard
     %----------------- End update
     %%%% Check stopping criteria, Distortion or mxmov2
     if Distnce==3 | hard_fuzzy==2
     mx_movcc=max(sum(clus_centr_new'-clus_centr_old').^2)
     JMD(m+1)=sum(sum((U.^Mpara).*Distance));
    if m<3, mx_movcc=mx_movcc+1e-6;end
     else
     Dist_old=Dist_new;
             Dist_new=sum(sum((U.^Mpara).*Distance));
     end  %if Dis
     %------------------ End check
  %%   Distance(1:row,asignd_class')
  %JMD(m+1)=sum(Distance(1:row,asignd_class'))
  m=m+1;
  lisdist=[lisdist;M m Dist_new mx_movcc];
  var=var+sign(mx_movcc-oldmxmov);
  list=[list;var reshape(clus_centr_new,1,M*col) reshape(pert,1,M*col)];
  oldmxmov=mx_movcc;
    if m==40
    [aa,bb]=min(list(:,1));
    clus_centr_new=reshape(list(bb,2:M*col+1),M,col);
    pert=reshape(list(bb,M*col+2:2*M*col+1),M,col);
    mx_movcc=0;
    end % if m



    %%%%%%%%%%%%% To display results in 2D case
    if plt==1
        if col>2,data2d=eigtrans(data_in,2);
else data2d=data_in;
end


if M<=6
plotclusters(data2d,M,asignd_class,A_inv,col,clus_centr_new,...
     Distnce,elps_cov,rsq,elps_cov,Which,savplt)
elseif M>6 & col==2
plotclusters2(data2d,M,asignd_class,A_inv,col,clus_centr_new,...
      Distnce,elps_cov,rsq,mn,mx,elps_cov,Which,savplt)
end %if M
    end %if col
    %----------- end display results
  end % end while D
  %----------------- End distortion calc loop
  if init_clus~=1,break,'p',end % quit in case of no splitting
end % end while M (splitting)
%------------- End main loop

%tot_time=etime(clock,tm)
tot_time=toc
```

```
itern=length(lisdist(:,1))
megaflops=flops/toc/1.e6
  if savres==1
  eval(['save ./autosaves/' int2str(Which) 'res clus_centr_new A_inv ...
lisdist tot_time asigned_class megaflops -ascii'])
  elseif savres==2
  eval(['save ./autosaves/' int2str(Which) 'res clus_centr_new A_inv ...
lisdist tot_time asigned_class megaflops'])
  end
  % leave space when you use save command (save ')

  %%%%% Iris data display
  if Data_choice==3
  [a,b]=max(U');
  b
  end
  %--------- End Iris

  % To measure the mutual MD between clusters
  % dist=zeros(codbksize,codbksize);
  for i=1:codbksize
    for j=1:codbksize
       if i~=j
       dist(i,j)=mah_dist(clus_centr_new(i,:),clus_centr_new(j,:),...
  reshape(A_inv(i,:),col,col));
       end
    end
  end
  %%% To run a movie of clustering until ctrl-c is pressed
  if col==2 & plt==1
%    while 1>0
for k=1:itern;
figure(k)
pause(1)
end
%    end
  end
  %-----------------
```

A.1.1 getdata.m. Any data set to be partitioned is stored in a specific subdirectory called **datasets** and included in the choices of the main program.

```
function data_in=getdata(Data_choice)
if Data_choice==1
load /home/hawkeye6/96d/96d/kyounis/dataset/xorData.asc
data_in=xorData(:,2:3);
elseif Data_choice==2
load /home/hawkeye6/96d/96d/kyounis/dataset/gaussians3
data_in=gaussians3;
  elseif  Data_choice==3
load /home/hawkeye6/96d/96d/kyounis/dataset/irisall
data_in=irisall(:,2:5);
  elseif Data_choice==4
load /home/hawkeye6/96d/96d/kyounis/dataset/daisy.train
% This ignores the first colm which is supposed to give class
jjj=length(daisy(1,:));
data_in=daisy(:,2:jjj);
elseif Data_choice==5
load /home/hawkeye6/96d/96d/kyounis/dataset/intens_vec
data_in=intens_vec;
elseif Data_choice==6
load /home/hawkeye6/96d/96d/kyounis/dataset/unif
data_in=unif(1:1000,:);
elseif Data_choice==7
load /home/hawkeye6/96d/96d/kyounis/dataset/flir.dat
data_in=flir(:,2:18);
elseif Data_choice==8
load /home/hawkeye6/96d/96d/kyounis/dataset/8dec44
data_in=intens_vec;
elseif Data_choice==9
```

```
load /home/hawkeye6/96d/96d/kyounis/dataset/s_shaped

data_in=s_shaped;

elseif Data_choice==10

load /home/hawkeye6/96d/96d/kyounis/dataset/linear2

data_in=lin_data;

else error('invalid choice')

end
```

A.1.2 KLI.m. This KLI initialization code implemented here was written by Dr. Martin DeSimio.

```
% kli.m
%
% Dr. Martin deSimio
% 22 august 1995
%
% use this routine to get an initial codebook of any size
% function klicbk = kli2(y,Ncdw);
% where
% y =    the set of data, one fv per row
%  Ncdw = the number of codewords to generate

function klicbk = kli2(y,Ncdw);

% make data set zero mean
% first find number of fv's (# rows)
ndrows = size(y,1);

meany = mean(y);
allmeany =  ones(ndrows,1)*meany;
y = y - allmeany;

% compute covariance of zero mean data, then find eigenvectors of
% covariance matrix
covy = cov(y);
[v,d] = eig(covy);

% find the total power
P = trace(d);
```

```
% get the number of dimensions
dims = size(covy,1);

% N(i) is the number of initial codewords along dimension i

N = zeros(dims,1);
for dimindx = 1:dims % for each dimension
   N(dimindx) = round( Ncdw * d(dimindx,dimindx)/P );
end
ncheck = sum(N);
if(ncheck ~= Ncdw)
  warning = 'not right number of codewords!'
  delta_ncdw = ncheck - Ncdw
  if (delta_ncdw > 0) % find dimension with most codewords and subtract
     [a,b] = max(N)
     N(b) = N(b) - delta_ncdw;
  end
  if(delta_ncdw < 0) % find dimenstion with least codewords and add
     [a,b] = min(N);
     N(b) = N(b) - delta_ncdw; % note that this will increase N(b)
  end
end
N;
newcheck=sum(N);


% now spread codewords out along eigenvectors;
% assume spread of +/- 2sigma; this keeps codwords within the data
row = 1;
```

```
for i = 1:dims
   if (N(i) ~=0 )
   sigm =  sqrt(d(i,i));
   if (rem( N(i),2))  == 0  % then even number of codewords
      oddity = 'no'; % in this dimension
      cell = 4*sigm/N(i);
      for j = 1:N(i)/2
         icbk(row,:) = v(:,i)'* (1/2)*(2*j -1)*cell;
         icbk(row+1,:) =  -1*icbk(row,:);
         row = row+2;
      end
   end

   if (rem( N(i),2)) == 1 % then odd number of codewords
    oddity = 'yes'; % in this dimension
      if (N(i) == 1)
        icbk(row,:) = sigm*0.001* randn(1,dims);
row = row+1; % added this fix on 26 july
      else
 icbk(row,:) = sigm*0.001*randn(1,dims);
 row = row+1;
         cell = 4*sigm/(N(i)-1);
         for j = 1: (N(i)-1)/2
            icbk(row,:) = v(:,i)'*j*cell;
    icbk(row+1,:) = -1*icbk(row,:);
            row = row+2 ;
         end
      end
   end
   end % for N(i) not = 0
```

```
end

% now add the mean value to the codebook vectors

addmn = ones(Ncdw,1)*meany;
klicbk = icbk + addmn;
```

A.1.3 plotclusters.m.

```
function plotclusters(data_in,M,asignd_class,A_inv,col,clus_centr_new,...
                      Distnce,elps_cov,rsq,elps_cov,Which,savplt)


mx_data=max(data_in);
mn_data=min(data_in);
mx=max(mx_data)+1;
mn=min(mn_data)-1;


figure('Position',[630, 700, 400 ,400])
hold on
ll1=find(asignd_class==1);
plot(data_in(ll1,1),data_in(ll1,2),'o')
ll2=find(asignd_class==2); plot(data_in(ll2,1),data_in(ll2,2),'x')
set(gca,'FontSize',8)
  if M==2, legend('Class 1','Class 2');
  elseif M>2,ll3=find(asignd_class==3);
  plot(data_in(ll3,1),data_in(ll3,2),'+')
    if M==3, legend('CLASS 1','CLASS 2','CLASS 3')
    elseif M>3,ll4=find(asignd_class==4);
    plot(data_in(ll4,1),data_in(ll4,2),'.')
if M==4,legend('Class 1','Class 2','Class 3','Class 4');
elseif M>4,ll5=find(asignd_class==5);
        plot(data_in(ll5,1),data_in(ll5,2),'ro')
          if M==5, legend('CLASS 1','CLASS 2','CLASS 3','CLASS 4','CLASS 5')
          elseif M>5,ll6=find(asignd_class==6);
          plot(data_in(ll6,1),data_in(ll6,2),'r+')
          legend('CLASS 1','CLASS 2','CLASS 3','CLASS 4','CLASS 5','CLASS 6')
          end
        end
    end
  end
% grid
axis([mn mx mn mx])
axis('square')
set(gca,'FontSize',15)
 if col==2
    for j=1:M
    elps_cov=reshape(A_inv(j,:),col,col);
    ellipse(clus_centr_new(j,:),elps_cov,rsq)
    xlabel('First Feature')
    ylabel('Second Feature')
    end %for j
  else
  xlabel('First Principal Component')
  ylabel('Second Principal Component')
  end %if col
set(gca,'FontSize',12);
hold off


  if savplt==1
  plt_name=['data' int2str(Which)];
  eval(['print -deps /home/hawkeye6/96d/96d/kyounis/PR/images/' plt_name])
  end %if savplt
```

A.2 Hyperellipsoidal Clustering (HEC) algorithm

Following are the MATLAB® codes used to implement the HEC algorithm. The code **HECVQ.m** implements the batch vector quantization algorithm described in HEC. Finally, the code **PCA.m** implements the neural network for principal component analysis.

A.2.1 HEC.m.

```
% HEC;
function class=HEC(Data_choice,codbksize);
%clear
close all
tic
flops(0)
con_flag=0;
plt_flag=1;
%%%%%%%%% load input data and substract the mean then normalize
%Data_choice= input('Data set (1) XOR data (2) Gaussians (3) IRIS (4) Daisy ...
                              (5) Mandrill (6) uniform (7) flir (8) Lenna ...
                              (9) sub-manifold  (10) linear : ');
%codbksize = input('Codebook size : ');
data_in=getdata(Data_choice);
[row,col]=size(data_in);
mn=mean(data_in);
data_in=data_in-ones(row,1)*mn;
max_length=max(diag(data_in*data_in'));
data_in=data_in./(ones(row,col)*sqrt(max_length));
mx_data=max(data_in);
mn_data=min(data_in);
mx=max(mx_data)+1;
mn=min(mn_data)-1;
lim1=[mn mx mn mx];
```

```
%------------------------------
%%%%%%% initialize
W=ones(codbksize,1)*reshape(eye(col),1,col^2);
%eval=ones(codbksize,col);
%deal with column vectors in cc
cc_old=0.1*randn(col,codbksize);
%cc_old=.1*[1 2;2 3 ;1 2;3 1];
V=cc_old;
Y=data_in;
lambda=0;
Delt_lambda=.2;
lambdamin=0;
epsl=1e-6;
S=ones(codbksize,col);
cons=ones(1,codbksize);
thresh=1e-4;
diff=1;
m=0;
oldclass=zeros(1,row);
inv_cov_mtx=ones(codbksize,1)*reshape(eye(col),1,col^2);
%--------- End initialization
  %%%%%%%%%%%%% Start main loop
  while diff>thresh
  [cc_new,class,B,mvq]=HECVQ(data_in,Y,cc_old,V,W,S,lambda,m,lim1,con_flag,...
                             cons,inv_cov_mtx);
    if Data_choice==3,class
    end %if data

    if col==2 & plt_flag==1
    h3=figure('Position',[1, 600, 400 ,400]);
```

```
     h4=figure('Position',[1, 100, 400 ,400]);
     end %if col

     %%%%%%%%%  compute new mean, cov according to the new partitioning
     for i=1:codbksize
     [val2,ind2]=find(class==i);
     len=length(ind2);
       if con_flag==1,cons(i)=len/row;end
       if len>1
%%%%%%%%% Get evec,eval,mean and covr
[trans,M,covr,vec,val]=eigtrans(data_in(ind2,:),2,0,0);
%-----------------------------------
cc_new(:,i)=M(:);
inv_cov=inv(covr);
inv_cov_mtx(i,:)=reshape(inv_cov,1,col^2);
W(i,:)=reshape(vec,1,col^2);
eval(i,:)=val;
S(i,:)=sqrt(val+epsl).^(-1);
V(:,i)=S(i,:)'.*(vec'*cc_new(:,i));
Y(ind2,:)=(ones(len,1)*S(i,:)).*(data_in(ind2,:)*vec);
else
'error'
  if len==1
  cc_new(:,i)=data_in(ind2,:)';
  else
  cc_new(:,i)=.01*randn(col,1);
  end %if len=1
end  %if length>1

%%%%%%%%  plot partitioning
```

```
if col==2 & plt_flag==1
figure(h3)
title(['m = ', int2str(m)])
  if i==1,     plot(data_in(ind2,1),data_in(ind2,2),'.')
  elseif i==2,plot(data_in(ind2,1),data_in(ind2,2),'x')
  elseif i==3,plot(data_in(ind2,1),data_in(ind2,2),'+')
  elseif i==4,plot(data_in(ind2,1),data_in(ind2,2),'o')
  end % if i==1
hold on
plot(cc_new(1,i),cc_new(2,i),'*')
ellipse(M,inv_cov,1)
end %if col
%----------End plotting
    end   %for i
    %---------- End computation
  if col==2 & plt_flag==1
  figure(h3),hold off
%  figure(h4),hold off
  end % if col
    if class==oldclass,break,else,oldclass=class;
    end %if class
  lambda=max(lambdamin,lambda-Delt_lambda);
  m=m+mvq
% if m==10,break,end
  cc_old=cc_new;
  end % while diff
  %---------- End main loop

tt=toc
megaflops=flops/tt/1.e6
```

```
%%%%%%%% Display results after rearranging

num_fig=gcf;

%while 1>0

%for i=1:num_fig

%figure(i),pause(1)

%end

%end
```

A.2.2 HECVQ.m.

```
function [cc_new,class,B,mvq]=HECVQ(norm_data,y,cc_old,V,evec,S,lambda,...
                                    mm,lim1,con_flag,cons,inv_cov_mtx)
[row,col]=size(norm_data);
%deal with column vectors in cc and V
codbksize=length(cc_old(1,:));
%%%%%%%% Initialization
B=0;
mvq=0;
diff=1;
thresh=1e-4;
plt_flag=0;
%---------------------

  %%%%%%%%%%%%%%% start the VQ loop
  while diff>=thresh
  cc_old;
    %%%%%%%%%%% Find new partitions (class) and projection of data points
    for i=1:row
    test=norm_data(i,:);
   %find distortions
     for j=1:codbksize
y1(j,:)= S(j,:).*(test*reshape(evec(j,:),col,col));
D(j)=cons(j)*(lambda*euclid(test',cc_old(:,j))+...
              (1-lambda)*euclid(y1(j,:)',V(:,j)));
end %for j
    [val,ind]=min(D);
    class(i)=ind;
    y(i,:)=y1(ind,:);
    end %for i
    %-----------------------------------------

    %%%%%%%%%%% compute the new means & their projection
    list=[];
    for i=1:codbksize
    [val2,ind2]=find(class==i);
    len(i)=length(ind2);
     if con_flag==1, cons(i)=max(len(i)/row,1/codbksize);
end % if con

if len(i)>1
      cc_new(:,i)=mean(norm_data(ind2,:))';
else
'one/no sample',i
list=[list i];
end %if len
    end %for i

  bad_cc=length(list);
  [dum1,dum2]=sort(len);
  big_clus=fliplr(dum2);

    for i=1:bad_cc
      if mm==0, pert=1e-2*randn(1,col)
      else
      pert=S(big_clus(i),:).^(-2)*(reshape(evec(i,:),col,col))'
      evec(list(i),:)=evec(big_clus(i),:);
      inv_cov_mtx(list(i),:)=inv_cov_mtx(big_clus(i),:);
      end %if mm
    cc_new(:,list(i))=cc_new(:,big_clus(i))-pert';
```

```
 cc_new(:,big_clus(i))=cc_new(:,big_clus(i))+pert';
 end % for i

 % compute the centroids in the whitened space
 for i=1:codbksize
 V(:,i)=S(i,:)'.*(reshape(evec(i,:),col,col)'*cc_new(:,i));
 end
 %------------------------------------------------

%%%%%%%%%%%%%% Plot partitions
if col==2 & plt_flag==1
h1=figure('Position',[500, 1, 700 ,700]);
%  h2=figure('Position',[820, 1, 300 ,700]);
 for i=1:codbksize
 [val2,ind2]=find(class==i);
 figure(h1)
 plot(cc_new(1,i),cc_new(2,i),'*')
 hold on
   if i==1,plot(norm_data(ind2,1),norm_data(ind2,2),'.')
   elseif i==2,plot(norm_data(ind2,1),norm_data(ind2,2),'x')
   elseif i==3,plot(norm_data(ind2,1),norm_data(ind2,2),'+')
   elseif i==4,plot(norm_data(ind2,1),norm_data(ind2,2),'o')
   end
 ellipse(cc_new(:,i),reshape(inv_cov_mtx(i,:),col,col),1)
 axis(lim1)
 end % for i
 figure(h1),  hold off
 end %if col
 %----------- End Plotting
```

```
  mvq=mvq+1
    if mvq==10;break,
    end
  diff=max(diag(euclid(cc_old,cc_new)));
  cc_old=cc_new;
  end %while diff
  %----------- End VQ loop
```

A.2.3 PCA.m.

```
clear

%%%%%%%%%%%%%%%%%%% load input data and substract the mean
Data_choice= input('Data set (1) XOR data (2) Gaussians (3) IRIS (4) Daisy ...
                              (5) Mandrill (6) uniform (7) flir (8) Lenna ...
                              (9) sub-manifold : ');
data_in=getdata(Data_choice);
[row,col]=size(data_in);
mn=mean(data_in);
norm_data=data_in-ones(row,1)*mn;

[vec,val]=eig(cov(norm_data));
[lambda,k]=sort(diag(val));
k=k(col:-1:1,1)';
vec=vec(:,k);
vec=vec./(ones(col,1)*max(abs(vec)))
lambda=lambda(col:-1:1,1)'

%%%%%%%%%%%%%%%%%%%% initialize
thresh=1e-2;
W=rand(col,col);
```

```
i=0;
H=zeros(1,col);
U=triu(rand(col,col),1);
eta=4e-2;
Mu=9e-2;
epoch=0;

  while max(max(abs(U)))>thresh
  i=i+1;
  pattern=norm_data(i,:)';
    if i==row,i=0;epoch=epoch+1,W./(ones(col,1)*max(abs(W))),end
    for j=1:col;
    H(j)=W(:,j)'*pattern+H*U(:,j);
    end % for j
  Delt_W=eta*pattern*H;
  Delt_U=-Mu*triu(H'*H,1);
  W=W+Delt_W;
  U=U+Delt_U;
  W=W./(ones(col,col)*max(max(abs(W)))); % normalize W
  if epoch==100,break,end
  end % while max
```

Appendix B. Vector Quantization Code

The following MATLAB® code is used to perform vector quantization coding of images.

B.1 ImageVQ.m

```
% Vector quantization
%function [Xnew,kh]=ImageVQ(Data_choice,codbksize,hard_fuzzy,Mpara,Distnce,init_clus)


tm=clock;
M=4; % codebook size
R=log(M)/log(2); %no. of bits to represent codebook ( M= 2^R )
%  if nargin<1
  Mpara=0;  % to set it zero if not fuzzy
  %%%%%%% Get parameters %%%%%%%%%%%%%%%
  Data_choice= input(' Data set (1) XOR data (2) Gaussians (3) IRIS (4)Daisy...
                                (5) Mandrill (6) uniform (7) flir (8) Lenna ...
                                (9) sub-manifold :');
  codbksize = input('Codebook size : ');
  hard_fuzzy = input('Clustering Algorithm (1) LBG ( HCM) (2) Fuzzy C_means : ');
  Distnce = input('Similarity Measure (1) Euclidean Distance ...
                                    (2) Diagonal covariance ...
     (3) changing Mahalanobis : ');
  init_clus = input('Codebook Generation  (1) Splitting (2) first C samples ...
                          (3) Minimax (4) Bezdek (5) Desimio (6) get-then-split : ');
   if hard_fuzzy==2
   mm = input('M parameter for Fuzzy C_means : ');
   end
%  end%if nargin
 %----------------------------------------------------
Which=[Data_choice codbksize hard_fuzzy Mpara Distnce init_clus]
```

```
%%%%%%%%  load the test image  %%%%%%%%%%%%%%%%%%%%%%%%%
clear dis_test
  if Data_choice==5
  load ./autosaves/mandrill
  [rom,com]=size(X);
  ival = mean(map')'; % the average of the 3 values of colormap (B&W intensity)
  newmap=[ ival ival ival];
    for i=1:rom
for j=1:com
X_intens(i,j)=ival(X(i,j));
% Matrix of 480x500 of intensity rather than indecies as in X
end
    end
  elseif Data_choice==8
  load ./autosaves/lenna
  X_intens=lenna;
  else error('invalid choice')
  end
[ro,co]=size(X_intens);
%-------------------------------------------------------


%%%%%%%%%%%%% Decimate the image  %%%%%%%%%%%%%%%%%%%%%%%
% Get upper-left pixel from 4x4 block, and and arrange each 4x4 of them in a row
% [intens_vec,x_intens]=decimate(X_intens,4,4);
% Or get saved decimated image directly
eval(['load /home/hawkeye6/96d/96d/kyounis/dataset/' int2str(Data_choice) 'dec44'])
%-----------------------------------------------------------------
```

```
%%%%%%%% Find codebook %%%%%%%%%%%%%%%%%%%%%%%
%[A2,lis2]=Cluster(intens_vec,M);
% Or you can load predetermined codebook (and A_inv matrix if needed)
% to load the codebook of the same image
load /home/hawkeye6/96d/96d/kyounis/PR/clustering/autosaves/8410352res
% to load a specific codebook
%load /home/hawkeye6/96d/96d/kyounis/PR/clustering/autosaves/541035res
%-------------------------------------------------------------

%%%%%%%% Perform VQ to reconstruct the image %%%%%%%%%
A2=clus_centr_new;
  for i=1:4:ro
    for j=1:4:co
    x_test=reshape(X_intens(i:i+3,j:j+3),1,16);
    diff=ones(M,1)*x_test-A2;
    dis_test=sum(diff.^2');
    [R,ind]=min(dis_test);
    Xnew(i:i+3,j:j+3)=reshape(A2(ind,:),4,4);
    end
  end

eval(['save ./autosaves/' int2str(Which) ' Xnew'])
%------------------------------------------------------------------


%%%%%%%%%%%%%% compute distortion / view image %%%%%%%%%%%%%%%%
dist=mean(mean((X_intens-Xnew).^2))
mmm=max(max(Xnew));
Xnew3=Xnew*65/mmm; %scale such that max grey level is 65
figure('Position',[750, 700, 300 ,300])
```

```
image(Xnew3)
axis off
colormap(gray)
plt_name=['recons' int2str(Which)]
eval(['print -deps  /home/hawkeye6/96d/96d/kyounis/PR/VQ/images/' plt_name])
%---------------------------------------------------------------

tot_time=etime(clock,tm)
```

Bibliography

1. Backer, E. and A. K. Jain. "A Clustering Performance Measure Based on Fuzzy Set Decomposition," *IEEE Trans. Pattern Anal. Machnine Intell.*, *PAMI-3*(1):66–74 (1981).
2. Ball, G. H. and D. G. Hall. "Some fundamental Concepts and Synthesis Procedures for Pattern Recognition Preprocessors," *In Proc. Int. Conf. Microwaves, Circuit Theory, Information Theory*, 281–297 (September 1964). Tokyo, Japan.
3. Baum, L., et al. "A maximization Technique Occuring in the Statistical Analysis of Probailistic Functions of Markov Chains," *Ann. Math Statistics*, 164–171 (1970).
4. Bezdek, James. *Pattern Recognition with Fuzzy Objective Function Algorithm.* Plenum Press, New York, 1981.
5. Bezdek, James, et al. "Some New Competitive Learning Schemes," *SPIE*, *2492* (1995).
6. Bezdek, James C. *Fuzzy Mathematics in Pattern Classification.* PhD dissertation, Cornell Univ., Ithaca, NY, 1973.
7. Bezdek, James C. *Self-Organization and Clustering Algorithms.* Technical Report, Defense Technical Information Center (DTIC) N91-21783, 1991.
8. Carpenter, G. and S. Grossberg. "Adaptive Resonance Theory: Stable Self Organization of Neural Recognition Code in Response to Arbitrary Lists of Input Patterns," *In Proc. 8th Annual Conf. Cognitive Science*, 45–62 (1986).
9. Darken, Christian and John Moody. *Fast Adaptive k-means Clustering: Some Imperical Results.* Technical Report, AFOSR Grant 89-0478, 1989.
10. DeSieno, D. "Adding Conscience to Competitive Learning," *IEEE Proceedings Int. Conf. in Neural Networks (ICNN-88)*, 117–124 (1988). San Deigo, CA.
11. DeSimio, Martin, "AFIT/EENG 817 Class Notes and Personal Interviews." The Air Force Institute of Technology, 1995.
12. Duda, Richard O. and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.
13. Dunn, J. C. "A Fuzzy Relative of the ISODATA Process," *Cybernetics*, 32–57 (March 1974).
14. Gath, I. and A. B. Geva. "Unsupervised Optimal fuzzy Clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *PAMI-11*(7):773–781 (July 1989).
15. Gersho, Allen and Robert M. Gray. *Vector Quantization and Signal Compression.* Boston, MA: Kluwer, 1992.
16. Gonzalez, Rafael C. and Paul Wintz. *Digital Image Processing* (2nd Edition). Addison Wesley, 1987.
17. Gray, Robert M. "Vector Quantization," *IEEE ASSP Magazine*, *1*:4–29 (1989).

18. Jain, A. K. and R. C. Dubes. *Algorithms for Clustering Data.* Englewood Cliffs, NJ: Prentice-Hall, 1988.

19. Jain, Anil K. and Jianchang Mao. *Computational Intelligence Imitating Life.* IEEE Press Marketing, 1994.

20. Jolion, J. M., et al. "Robust Clustering with Applications in Computer Vision," *IEEE Trans. Pattern Anal. Machine Intell., PAMI-13*(8):791–802 (1991).

21. Katsavounidis, Ioannis, et al. "A New Initialization Technique for Generalized Lloyd Iteration," *IEEE Signal Processing Letters, 1*(10):144–146 (October 1994).

22. Keller, James M., et al. "A Fuzzy K-Nearest Neighbor Algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, SMC-15*(4):580–585 (July/August 1985).

23. Kohonen, Tuevo T. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 1989.

24. Linde, Y., et al. "An Algorithm for Vector Quantizer Design," *IEEE Trans. Commun., COM-28*:84–95 (1980).

25. Mao, Jianchang and Anil K. Jain. "A Self-Organizing Network for Hyperellipsoidal Clustering," *IEEE Transactions on Neural Networks, 7*(1):16–29 (January 1996).

26. McQueen, J. B. "Some Methods of Classification and Analysis of Multivariate Observations," *In Proc. 5th Berkeley Symp. Math. Statis. Probability*, 281–297 (1967).

27. Patrick, Edward A. and Leon Y. L. Shen. "Interactive Use of Problem knowledge for Clustering and Decision Making," *IEEE Tran. on Computers*, 216–222 (February 1971).

28. Reynolds, Douglas A. *A Gaussian Mixture Modeling Approach to Text-Independent Speaker Identification.* PhD dissertation, Georgia Institute of Technology, August 1992.

29. Rogers, Steven K., et al. *An Introduction to Biological and Artificial Neural Networks.* Bellingham, Washington: SPIE Optical Engineering Press, 1991.

30. Rubner, J. and P. Tavan. "A Self Organizing Network for Principal Component Analysis," *Europhysics letters, 10*:693–698 (1989).

31. Ruck, Dennis W. and Steven Rogers. "The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function," *IEEE Transactions on Neural Networks, 1*(2):296–298 (December 1990).

32. Ruck, Dennis W., et al. "The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function," *IEEE Transactions on Neural Networks* (1990).

33. Tou, Julius T. and Rafael C. Gonzalez. *Pattern Recognition Principles.* Reading, MA: Addison-Wesley Publishing Company, 1974.

34. Windham, M. P. "Cluster Validity of the Fuzzy c-means Clustering Algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-4*(4):357–363 (1982).

35. Younis, Khaled S., et al. "Vector Quantization Based on Dynamic Adjustment of Mahalanobis Distance," *IEEE National Aerospace and Electronics Conference (NAECON), 2*:858–862 (May 1996).

36. Zadeh, Lotfi A. "Fuzzy Sets," *Information and Control*, *8*:338–353 (1965).

Vita

Captain Khaled S. Younis [redacted] He graduated First in the 1986 high school class at Dirar Ibnel Azwar School in Amman, Jordan. In 1990, he graduated from Mu'tah University with a Bachelor of Science Degree in Electrical Engineering and he was awarded a Royal prize granted for the first of the Engineering graduates. Upon graduation, Capt. Younis received his commission as a Second Lieutenant in the Royal Jordanian Air Force (RJAF). In 1994, he was selected by the RJAF to earn his Master of Science Degree in Electrical Engineering at the Air Force Institute of Technology (AFIT) at Wright-Patterson AFB, Ohio. Capt Younis was awarded a scholarship by the Dayton Area Graduate Studies Institute (DAGSI) to pursue a Doctorate of Philosophy Degree in Electrical Engineering after completion of his Master's program at AFIT.

[redacted]

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 1996	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE WEIGHTED MAHALANOBIS DISTANCE FOR HYPER-ELLIPSOIDAL CLUSTERING	**5. FUNDING NUMBERS**
6. AUTHOR(S) Khaled S. Younis Captain, RJAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology WPAFB OH 45433-6583	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/96D-22
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Major Thomas Burns, WL/AARA 2010 fifth Street, Bldg. 23 WPAFB, OH 45433	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; Distribution Unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

Abstract

Cluster analysis is widely used in many applications, ranging from image and speech coding to pattern recognition. A new method that uses the weighted Mahalanobis distance (WMD) via the covariance matrix of the individual clusters as the basis for grouping is presented in this thesis. In this algorithm, the Mahalanobis distance is used as a measure of similarity between the samples in each cluster. This thesis discusses some difficulties associated with using the Mahalanobis distance in clustering. The proposed method provides solutions to these problems. The new algorithm is an approximation to the well-known expectation maximization (EM) procedure used to find the maximum likelihood estimates in a Gaussian mixture model. Unlike the EM procedure, WMD eliminates the requirement of having initial parameters such as the cluster means and variances as it starts from the raw data set. Properties of the new clustering method are presented by examining the clustering quality for codebooks designed with the proposed method and competing methods on a variety of data sets. The competing methods are the Linde-Buzo-Gray (LBG) algorithm and the Fuzzy *c*-means (FCM) algorithm, both of them use the Euclidean distance. The neural network for hyperellipsoidal clustering (HEC) that uses the Mahalnobis distance is also studied and compared to the WMD method and the other techniques as well. The new method provides better results than the competing methods. Thus, this method becomes another useful tool for use in clustering.

14. SUBJECT TERMS Clustering, Unsupervised Learning, Mahalanobis Distance, Neural Networks, Pattern Recognition	**15. NUMBER OF PAGES** 109
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL